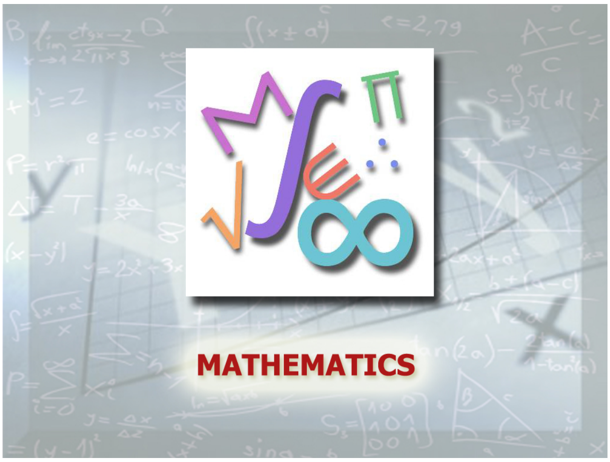
Università degli Studi di Torino
Dipartimento di Matematica

Scuola di Dottorato in Scienza ed Alta Tecnologia

Ciclo XXV



Interactive Realizability, Monads and Witness Extraction (Draft)

Giovanni Birolo

Tutor: Prof. Stefano Berardi

Coordinatore del Dottorato: Prof. Luigi Rodino

Anni Accademici: 2010–2012

Settore Scientifico-disciplinare di afferenza: MAT/01 Logica Matematica

Università degli Studi di Torino
Scuola di Dottorato in Scienza ed Alta Tecnologia

Tesi di Dottorato di Ricerca in Scienza ed Alta Tecnologia
Indirizzo: Matematica

INTERACTIVE REALIZABILITY, MONADS AND WITNESS EXTRACTION (DRAFT)

Giovanni Birolo

Tutor: Prof. Stefano Berardi

XXV ciclo – Aprile 2012

Abstract

In this dissertation we collect some results about "interactive realizability", a realizability semantics that extends the Brouwer-Heyting-Kolmogorov interpretation to (sub-)classical logic, more precisely to first-order intuitionistic arithmetic (Heyting Arithmetic, HA) extended by the law of the excluded middle restricted to Σ^0_1 formulas (EM_1), a system motivated by its interest in proof mining. These results are three interconnected works, listed below.

- We describe the interactive interpretation of a classical proof involving real numbers. The statement we prove is a simple but non-trivial fact about points in the real plane. The proof employs EM_1 to deduce properties of the ordering on the real numbers, which is undecidable and thus problematic from a constructive point of view.

- We present a new set of reductions for derivations in natural deduction that can extract witnesses from closed derivations of simply existential formulas in $\mathsf{HA} + \mathsf{EM}_1$. The reduction we present are inspired by the informal idea of learning by making falsifiable hypothesis and checking them, and by the interactive realizability interpretation. We extract the witnesses directly from derivations in $\mathsf{HA} + \mathsf{EM}_1$ by reduction, without encoding derivations by a realizability interpretation.

- We give a new presentation of interactive realizability with a more explicit syntax. We express interactive realizers by means of an abstract framework that applies the monadic approach used in functional programming to modified realizability, in order to obtain less strict notions of realizability that are suitable to classical logic. In particular we use a combination of the state and exception monads in order to capture the learning-from-mistakes nature of interactive realizers.

Contents

Chapter 1

Preface

1.1 Proofs and Computations

From the beginning intuitionistic logic has been linked to the idea of computation. In hindsight, this is already implicit in the Brouwer-Heyting-Kolmogorov (BHK) interpretation, which is presented in terms of proofs, constructions and transformations thereof (or problems in the case of Kolmogorov).

The connection becomes more evident with the introduction of recursive realizability by Kleene in [14] and, later, modified realizability by Kreisel in [15]. Realizability semantics can be thought of as formalizations of the BHK interpretation, where the vague notions of proof, construction and transformation are replaced with the notions of computable functionals.

The full explicitation of this connection is the Curry-Howard correspondence, [13], where the whole proof is seen as a program and the conclusion as the type or the specification of the program. While interpreting an intuitionistic proof as a computation is quite natural (in hindsight), this is not the case for classical proofs.

A computational interpretation of a classical proof can be obtained by first translating a classical proof into an intuitionstic one by means of double-negation translation. This approach was used by Gödel to prove relative consistency results for classical and intuitionistic arithmetic. However, the double-negation translation transforms informative statements into non-informative ones, so the computations we can extract in this way yield trivial results.

Moreover, this approach is indirect, while proofs and computations are almost undistinguishable in intuitionstic logic.

We quote from [23]:

> Until around 1990 there was a widespread consensus to the effect that "there is no Curry-Howard isomorphism for classical logic." However, at that time Tim Griffin made a path-breaking discovery which have convinced most critics that classical logics have something to offer the Curry-Howard isomorphism.

In [12], Griffin extends the Curry-Howard correspondence to classical proofs, employing functional programs with first-class continuations. In Griffin's own words:

> The programming language Scheme contains the control construct call/cc that allows access to the current continuation (the current control context). This, in effect, provides Scheme with first-class labels and jumps. We show that the well-known formulae-as-types correspondence, which relates a constructive proof of a formula α to a program of type α, can be extended to a typed Idealized Scheme. What is surprising about this correspondence is that it relates classical proofs to typed programs.

After Griffin's discovery, other interpretations extending the Curry-Howard correspondence to classical logic have been put forward. In [19], Parigot introduces the $\lambda\mu$-calculus, an extension of lambda calculus with an additional kind of variables for subterms.

In [16], Krivine devised a new notion of realizability for classical logic called "classical realizability". In classical realizability realizers are written in an untyped lambda calculus with save/restore operators for the execution context and they are interpreted by an abstract machine that allows the manipulation of execution contexts, represented as "stacks" of arguments.

Interactive realizability is a more recent proof interpretation for classical logic and the main focus of this dissertation.

1.2 Interactive Realizability

Introduced by Berardi and de'Liguoro in [5, 6], interactive realizability is a technique for understanding and extracting the computational content in the case of the sub-classical logic

$\mathsf{HA} + \mathsf{EM}_1$ (Heyting Arithmetic extended by the law of the excluded middle restricted to Σ^0_1 formulas).

The main inspiration sources for interactive realizability are Coquand's game theoretic semantics for classical arithmetic and Gold's idea of learning in the limit.

Gold original interest is language learnability, for instance a child learning the grammar of a language by repeated exposure to correct sentences. We expect that the child will eventually learn the language and stop making mistakes when speaking. The interesting point is that we do not know how many sentences he needs to complete the learning, In [11], he defines what it means to learn the answer of some question from an unlimited amount of evidence and in a finite time as follows:

> The purpose of this paper is to discuss the classes of problems that can be solved by infinitely long decision procedures in the following sense: An algorithm is given which, for any problem of the class, generates an infinitely long sequence of guesses. The problem will be said to be solved in the limit if, after some finite point in the sequence, all the guesses are correct and the same (in case there is more than one correct answer).

In [9], Coquand presents a novel game theoretic semantics. As customary in game semantics, each formula defines a game for two players: ∃loise, trying to show that the formula is true and ∀belard, trying to show that it is not. A formula is then validated by the existence of a winning strategy for ∃loise.

Coquand takes the game for intuitionistic logic and extends it to classical logic by allowing ∃loise to retract her moves: instead of answering to the last move made by ∀belard, she can change her mind on her previous moves and go back to any past position. Thus a new game with asymmetric backtracking is defined, where ∃loise holds the advantage and the existence of a backtracking strategy validates classical logic.

In [5, 6], Berardi and de'Liguoro recast Coquand's idea of backtracking strategy as a strategy for learning the truth of classical statements in the limit in Gold's sense. Moreover, they present their proof interpretation as a realizability rather than game theoretic semantics and write backtracking strategies as learning algorithms in a simply typed λ-calculus with primitive recursion.

The aim of interactive realizability as a proof interpretation for classical logic is to ex-

press the computational content of a suitable subset of classical proofs in an *understandable* form. This is the motivation for its peculiar features, which we summarize in the following.

- The interactive interpretation is "faithful" to the classical proof, meaning that the computation follows closely the original proof. This is possible since interactive realizability interprets the proof directly, without resorting to proof translations. We also avoid adding computations that are not explicitly present in the proof, for instance blind searches to realize existential statements.

- A common feature of computational interpretation of classical logic is that they extract programs that manipulate the execution context, that is, they need continuations. However, the use of continuations can make a program hard to follow. Interactive realizability uses the idea of learning to explain the manipulations of the execution contexts that are needed to backtrack. In particular this is accomplished by means of a knowledge state, that is increased during the learning process and that act as a guide in the exectution of the interactive interpretation.

- Interactive realizability is compositional, meaning that the interactive interpretations of different parts of a single proofs can be given independently and then composed to obtain the interactive interpretation of the whole proof.

- In this dissertation we only consider proofs where the law of the excluded middle is restricted to Σ_1^0 formulas. In this case interactive realizers use simpler constructs like states and exceptions instead of continuations in order to handle the backtracking nature of the computational content of classical proofs.

Chapter 2

Preliminaries

In this chapter we introduce the notation and the tools we shall in the rest of the thesis. Our analysis is mainly concerned with proofs in first-order arithmetic, both intuitionistic (HA) and classical (PA). We also introduce a simply typed lambda calculus, which we use to give realizability interpretations of proofs.

2.1 Constructive Arithmetic in Natural Deduction

In this section, we introduce Heyting Arithmetic and the axiom for the law of the excluded middle, which will be the logical setting of the whole dissertation. We briefly describe the language of first-order logic, the rules of minimal logic in natural deduction, the axioms and rules of arithmetic and the restricted excluded middle axiom schemes.

2.1.1 Primitive Recursive Functions and Predicates

In the language of arithmetic we include symbols for all the primitive recursive functions and predicates in arithmetic. We briefly recall their definition.

We only consider arithmetical functions, that is, functions from the natural numbers to the natural numbers, which we denote with $\mathbb{N}$. These functions take n arguments for some natural number n and are called n-ary. We use the metavariables $f^{(n)}, g^{(n)}, h^{(n)}$ for n-ary functions.

Primitive recursive functions are defined by induction. The basic primitive recursive functions are the following:

the constant function the 0-ary constant function $\mathbf{0}$,

the successor function the 1-ary successor function succ, which returns the successor of its argument,

the projection functions for every $n \geq 1$ and each i with $1 \leq i \leq n$, the n-ary projection function P^n_i, which returns its i-th argument:

$$P^n_i(x_1, \dots, x_n) = x_i.$$

More complex primitive recursive functions can be obtained by combining basic or previously defined primitive recursive function. Given the primitive recursive functions $g^{(n)}$, $h^{(m)}_1, \dots, h^{(m)}_k$ and $h^{(k+2)}$, we can define new primitive recursive functions in two ways:

composition the composition of g with $h_1, \dots, h_n$, i.e. the m-ary function:

$$comp(g, h_1, \dots, h_n) = g(h_1(x_1, \dots, x_m), \dots, h_n(x_1, \dots, x_m)),$$

is primitive recursive;

primitive recursion the $(k+1)$-ary function $recgh$ is defined as the primitive recursion of g and h, i.e. the function:

$$recgh(0, x_1, \dots, x_k) = g(x_1, \dots, x_k),$$
$$recgh(\mathsf{succ}(y), x_1, \dots, x_k) = h(y, rec(y, x_1, \dots, x_k), x_1, \dots, x_k),$$

is primitive recursive.

Now we can define primitive recursive predicates by saying that they are the predicates whose characteristic function is a primitive recursive function. More precisely an n-ary predicate p is primitive recursive if and only if there is a primitive recursive $f^{(n)}$ such that:

$$p(x_1, \dots, x_n) \text{ if and only if } f(x_1, \dots, x_n) = 1,$$

for any $x_1, \dots, x_n \in \mathbb{N}$.

2.1.2 The Language of Arithmetic

In this subsection we define the language of first-order arithmetic.

Let $\{\mathcal{F}_n\}_{n\in\mathbb{N}}$ and $\{\mathcal{R}_n\}_{n\in\mathbb{N}}$ be two indexed sets of non-logical symbols. We assume that $\mathcal{F}_n$ contains symbols for all and only the primitive recursive (total) functions of arity n, that is, functions in $\mathbb{N}^n \to \mathbb{N}$. Compare our language with the language of Primitive Recursive Arithmetic (PRA) [22]. Since we have induction on quantified formulas, unlike in PRA, in principle we only need to define zero constant, the successor function, addition and multiplication (see [17, p. 155]). However, we prefer a richer language with symbols for all the recursive functions and relations, because it is simpler to use.

Similarly we assume that $\mathcal{R}_n$ contains symbols for all and only the primitive recursive relations of arity n, that is, subsets of $\mathbb{N}^n$. We use the metavariables $f^{(n)}, g^{(n)}, h^{(n)}$ for function symbols and $p^{(n)}, q^{(n)}, r^{(n)}$ for relation symbols, omitting the superscript when we do not need it.

The 0-ary symbols are called *constants*. We assume that some standard symbols are present:

$\mathcal{F}_0$	$\mathcal{F}_1$	$\mathcal{F}_2$	$\mathcal{R}_0$	$\mathcal{R}_2$
$\mathbf{0}$	succ	$+, \cdot$	$\top, \bot$	$=, <, \leq$

For the sake of readability we informally write $\boldsymbol{n}$ instead of $\mathsf{succ}^n\,\mathbf{0}$ and we shall use the infix notation for binary functions and relations.

Let $\mathcal{V}$ be an enumerable set of variable symbols. We use the metavariables x, y, z for variable symbols.

We use the metavariable t for arithmetic terms, which are defined as:

$$t ::= x \mid f^{(n)}(t_1, \ldots, t_n)$$

We use the metavariables P, Q, R for atomic formulas, defined as:

$$P ::= p^{(n)}(t_1, \ldots, t_n)$$

Finally, we use the metavariables A, B, C for (well formed) formulas, defined as:

$$A ::= P \mid B \wedge C \mid B \vee C \mid B \to C \mid \forall x.\, B \mid \exists x.\, B$$

The entire grammar is given more concisely in Table 2.1.

Table 2.1: The language of first-order arithmetic.

	Metavariables	Definition
function symbols	$f^{(n)}, g^{(n)}, h^{(n)}$	elements of $\mathcal{F}_n$
relation symbols	$p^{(n)}, q^{(n)}, r^{(n)}$	elements of $\mathcal{R}_n$
arithmetic variables	x, y, z	elements of $\mathcal{V}$
arithmetic terms	t	$t ::= x \mid f^{(n)}(t_1, \ldots, t_n)$
atomic formulas	P, Q, R	$P ::= p^{(n)}(t_1, \ldots, t_n)$
formulas	A, B, C	$A ::= P \mid B \wedge C \mid B \vee C \mid$ $\mid B \to C \mid \forall x.\ B \mid \exists x.\ B$

We write $\clubsuit[x_1, \ldots, x_n := t_1, \ldots, t_n]$ for the *simultaneous substitution* of the variables $x_1, \ldots, x_n$ with the terms $t_1, \ldots, t_n$ in the expression $\clubsuit$ (a term or a formula).

We use a compact notation for bounded quantification on natural numbers:

$$\forall x \leq t.\ .A \text{ stands for } \forall x.\ x \leq t \to A,$$

$$\exists x \leq t.\ .A \text{ stands for } \exists x.\ x \leq t \wedge A.$$

The language of first-order arithmetic is the language of both Heyting Arithmetic and Peano Arithmetic.

2.1.3 Reduction on Arithmetic Terms

In this subsection we introduce a reduction on arithmetic terms. Since arithmetic terms are build from recursive primitive functions, we can transform the equations defining them into reductions in a natural way.

A term t is a *numeral* if it is either $\mathbf{0}$ or succ^u for some numeral u. We consider numerals as the basic arithmetic terms, so we do not reduce them.

Consider a term t build by composition, for instance:

$$f(t_1, \ldots, t_i, \ldots, t_n).$$

The first option is to reduce one of the arguments: for any $0 < i \leq n$, if t_i reduces to t_i', then

$$f(t_1, \ldots, t_i, \ldots, t_n) \to f(t_1, \ldots, t_i', \ldots, t_n).$$

The second option is to reduce the whole term. In this case which reduction we use depends on how the primitive recursive function denoted by f is defined. As we said, if f denotes zero or the successor function, it does not reduce. If f denotes a projection function P^n_i, then it reduces as follows:

$$f(t_1, \ldots, t_n) \to t_i,$$

exactly as in the definition of P^n_i. If f denotes the composition of functions whose symbols are $g, h_1, \ldots, h_n$, then it reduces as follows:

$$comp(g, h_1, \ldots h_n) = g(h_1(t_1, \ldots, t_m), \ldots, h_n(t_1, \ldots, t_m)).$$

If f the primitive recursion of two other functions whose symbols are g and h, then f reduces as follows:

$$f(\mathbf{0}, t_1, \ldots, t_k) \to g(t_1, \ldots, t_k),$$
$$f(\mathsf{succ}(u), t_1, \ldots, t_k) \to h(u, rec(u, t_1, \ldots, t_k), t_1, \ldots, t_k),$$

depending on the form of the first argument.

We basically described how to compute primitive recursive functions. This reduction is strongly normalizing and the normal form is unique. In the case of closed term the normal form is a numeral.

We can extend this reduction from terms to formulas, by reducing the terms contained in a formula. Strong normalization and uniqueness are preserved.

2.1.4 Axioms and Rules of Intuitionistic Logic

In this subsection we describe natural deduction as a notation for formal proofs and the axioms and the rules of intuitionistic logic with equality.

A *derivation* is a formal diagram that describes a proof. We write derivations in natural deduction, that is, as labeled trees of annotated formulas, with the requirement that the any subtree conforms to one of a number of patterns called *rules of inference* or simply *rules*. An annotated formula consists of a formula, the name of the rule it conforms to and an unique label. In a proof tree the same formula can and often does appear in more than one place. Similarly a rule can be applied many times. In order to distinguish these multiple instance,

when reasoning about the structure of a derivation we speak about *formula occurrences* and *rule instances*.

We draw derivations with the root at the bottom. The top formulas of the tree are axiom instances if they are derived from a rule with no premisses, discharged assumption instances if they are discharged by a rule instance below them or open assumption instances if they are not.

In an elimination rule the premiss containing the logical operator being eliminated is called the *major* premiss; the other premisses are called *minor* premisses. We follow a common convention in writing rules and place the major premiss in the leftmost position. For introduction and atomic rules we consider all premisses to be major premisses.

We call a rule *atomic* when its assumptions and conclusion are all atomic formulas. We call *atomic axiom* an atomic rule without premisses.

There are two notations for natural deduction that differ on how they represent open assumptions. For instance, consider the implication introduction rule written in two ways:

$$\rightarrow \mathrm{I}\,\frac{\begin{array}{c}[A]^{\alpha}\\ \vdots\\ B\end{array}}{A \rightarrow B}\,\alpha \qquad\qquad \rightarrow \mathrm{I}\,\frac{\Gamma, \alpha : A \vdash B}{\Gamma \vdash A \rightarrow B}\,\alpha$$

We say that the leftmost rule is written in Gentzen's style and the rightmost one in sequent style. In the Gentzen's style rule the open assumption A is inside square brackets with a superscript label α to show that it is discharged by the rule also labeled with α. Note that labels are essential: if we remove them we may not know which rule instance discharges an occurrence of an open assumption.

In the sequent style rule all the open assumptions a formula depends on are in the list (metavariable Γ) of couples of labels and formulas that precedes the symbol $\vdash$. The fact that the rule discharges an open assumption A is clear from the fact that A is in the open assumption list of the premiss and is not in the list of the conclusion.

The simplest rule is the identity rule, that shows how to use an open assumption: they say that if we assumed a formula A we can derive A, whence the name. When using Gentzen's style it is usually left implicit, but it needs to be written in sequent style. We give both versions:

$$\mathrm{Id}\,\frac{[A]^{\alpha}}{A}, \qquad \mathrm{Id}\,\frac{}{\Gamma \vdash A},$$

where Γ contains $\alpha : A$ for some label α.

We can now present the rules of minimal logic, a subset of intuitionistic and of classical logic. It consists of ten rules, one introduction rule and one elimination rule for each of the five logical connectives. They are listed in Gentzen's style in Figure 2.1 and in sequent style in Figure 2.2. The usual restrictions apply to rules $\forall$I and $\exists$E: in $\forall$I we assume that x is not free in any assumption in Γ; in $\exists$E we assume that $x \equiv y$ or y is not free in A and y is not free in any assumption in Γ.

Figure 2.1: Rules for minimal logic, in Gentzen's style natural deduction.

$$\wedge\mathrm{I}\ \frac{A \qquad B}{A \wedge B} \qquad\qquad \wedge\mathrm{E_R}\ \frac{A \wedge B}{A} \qquad \wedge\mathrm{E_L}\ \frac{A \wedge B}{B}$$

$$\vee\mathrm{I_R}\ \frac{A}{A \vee B} \qquad \vee\mathrm{I_L}\ \frac{B}{A \vee B} \qquad\qquad \begin{array}{l}\vee\mathrm{E_L}\\ \alpha\end{array}\ \dfrac{A \vee B \qquad \begin{array}{c}[A]^\alpha\\ \vdots\\ C\end{array} \qquad \begin{array}{c}[B]^\alpha\\ \vdots\\ C\end{array}}{C}$$

$$\begin{array}{l}\to \mathrm{I}\\ \alpha\end{array}\ \dfrac{\begin{array}{c}[A]^\alpha\\ \vdots\\ B\end{array}}{A \to B} \qquad\qquad \to \mathrm{E}\ \frac{A \to B \qquad A}{B}$$

$$\forall\mathrm{I}\ \frac{A}{\forall x.\, A} \qquad\qquad \forall\mathrm{E}\ \frac{\forall x.\, A}{A[x := t]}$$

$$\exists\mathrm{I}\ \frac{A[x := t]}{\exists x.\, A} \qquad\qquad \begin{array}{l}\exists\mathrm{E}\\ \alpha\end{array}\ \dfrac{\exists x.\, A \qquad \begin{array}{c}[A[x := y]]^\alpha\\ \vdots\\ C\end{array}}{C}$$

Intuitionistic logic has all the rules of minimal logic with the addition of the *ex falso quodlibet* rule:

$$\bot\mathrm{E}\ \frac{\bot}{A}$$

which can be thought of as an elimination rule for the atomic formula $\bot$. For technical reason we prefer to have atomic rules when possible, so instead of the $\bot$E rule we consider

Figure 2.2: Rules for minimal logic, in sequent style natural deduction.

$$\wedge\mathrm{I}\ \frac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \wedge B} \qquad \wedge\mathrm{E_R}\ \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A} \qquad \wedge\mathrm{E_L}\ \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash B}$$

$$\vee\mathrm{I_R}\ \frac{\Gamma \vdash A}{\Gamma \vdash A \vee B} \qquad \vee\mathrm{I_L}\ \frac{\Gamma \vdash B}{\Gamma \vdash A \vee B}$$

$$\vee\mathrm{E_L}\ \frac{\Gamma \vdash A \vee B \qquad \Gamma, \alpha_{k+1} : A \vdash C \qquad \Gamma, \alpha_{k+1} : B \vdash C}{\Gamma \vdash C}$$

$$\to \mathrm{I}\ \frac{\Gamma, \alpha_{k+1} : A \vdash B}{\Gamma \vdash A \to B} \qquad \to \mathrm{E}\ \frac{\Gamma \vdash A \to B \qquad \Gamma \vdash A}{\Gamma \vdash B}$$

$$\forall\mathrm{I}\ \frac{\Gamma \vdash A}{\Gamma \vdash \forall x.\, A} \qquad \forall\mathrm{E}\ \frac{\Gamma \vdash \forall x.\, A}{\Gamma \vdash A[x := t]}$$

$$\exists\mathrm{I}\ \frac{\Gamma \vdash A[x := t]}{\Gamma \vdash \exists x.\, A} \qquad \exists\mathrm{E}\ \frac{\Gamma \vdash \exists x.\, A \qquad \Gamma, \alpha_{k+1} : A[x := y] \vdash C}{\Gamma \vdash C}$$

its restricted version, where the conclusion can only be an atomic formula:

$$\bot\mathrm{E}_0\ \frac{\bot}{P}$$

Then $\bot\mathrm{E}_0$ is an atomic rule. The $\bot\mathrm{E}$ rule is admissible given the $\bot\mathrm{E}_0$ rule. It is easy to prove by induction on the structure of the conclusion of the $\bot\mathrm{E}$ rule. For instance we can prove $A = P \to Q$ from $\bot$:

$$\to \mathrm{I}\ \frac{\bot\mathrm{E}_0\ \dfrac{\bot}{Q}}{P \to Q}$$

First-order logic always assumes the existence of a binary relation symbol = and axioms and rules defining it as an equivalence relation compatible with functions and relations. These axioms and rules are given in Gentzen's style in Figure 2.3 and in sequent style in Figure 2.4. Note that they are all atomic.

Figure 2.3: Rules for the equality predicate, in Gentzen's style natural deduction.

$$\text{Refl}\ \frac{}{t = t} \qquad \text{Sym}\ \frac{t = u}{u = t} \qquad \text{Trans}\ \frac{t = u \qquad u = v}{t = v}$$

$$\text{Sub}_{\mathcal{F}}\ \frac{t_1 = u_1 \quad \dots \quad t_n = u_n}{f^{(n)}(t_1, \dots, t_n) = f^{(n)}(u_1, \dots, u_n)}$$

$$\text{Sub}_{\mathcal{R}}\ \frac{t_1 = u_1 \quad \dots \quad t_n = u_n \qquad p^{(n)}(t_1, \dots, t_n)}{p^{(n)}(u_1, \dots, u_n)}$$

Figure 2.4: Rules for the equality predicate, in sequent style natural deduction.

$$\text{Refl}\ \frac{}{\Gamma \vdash t = t} \qquad \text{Sym}\ \frac{\Gamma \vdash t = u}{\Gamma \vdash u = t} \qquad \text{Trans}\ \frac{\Gamma \vdash t = u \qquad \Gamma \vdash u = v}{\Gamma \vdash t = v}$$

$$\text{Sub}_{\mathcal{F}}\ \frac{\Gamma \vdash t_1 = u_1 \quad \dots \quad t_n = u_n}{\Gamma \vdash f^{(n)}(t_1, \dots, t_n) = f^{(n)}(u_1, \dots, u_n)}$$

$$\text{Sub}_{\mathcal{R}}\ \frac{\Gamma \vdash t_1 = u_1 \quad \dots \quad t_n = u_n \qquad \Gamma \vdash p^{(n)}(t_1, \dots, t_n)}{\Gamma \vdash p^{(n)}(u_1, \dots, u_n)}$$

2.1.5 Axiom and Rules of Arithmetic

In this subsection we present the axioms and the rules of Heyting Arithmetic (HA).

The rules defining the functions symbols succ, $+$ and $\cdot$ are in Figure 2.5.

Figure 2.5: Axioms and rules for the successor, addition and multiplication, in Gentzen's style natural deduction.

$$\text{Zero}\ \frac{\mathsf{succ}\, t = \mathbf{0}}{\bot} \qquad \text{Succ}\ \frac{\mathsf{succ}\, t = \mathsf{succ}(u)}{t = u}$$

$$\text{Add}_0\ \frac{}{t + \mathbf{0} = t} \qquad \text{Add}_{\mathsf{succ}}\ \frac{}{t + \mathsf{succ}(u) = \mathsf{succ}(t) + u}$$

$$\text{Mult}_0\ \frac{}{t \cdot \mathbf{0} = \mathbf{0}} \qquad \text{Mult}_{+}\ \frac{}{t \cdot \mathsf{succ}(u) = t \cdot u + t}$$

Induction

The last rule we need to add to have the full HA is induction. Induction can be thought of as the requirement that any natural number can be written as $\mathsf{succ}(\ldots(\mathsf{succ}(\mathbf{0})))$. In first-order logic is usually expressed by saying that, for any formula A, if $A(0)$ holds and $A(x+1)$ holds whenever $A(x)$ holds, then A holds for any natural number. Induction has many different but equivalent formulations and it can be written as either an axiom or a rule. The most common formulation is the following induction axiom schema:

$$A[x := \mathbf{0}] \wedge (\forall x.\, A \to A[x := \mathsf{succ}(x)]) \to \forall x.\, A,$$

where A is any formula. This axiom can be written as the *induction rule* in Gentzen's style:

$$\mathrm{Ind}\ \frac{A[x := \mathbf{0}] \qquad \begin{array}{c}[A]^{\alpha}\\ \vdots \\ A[x := \mathsf{succ}(x)]\end{array}}{\forall x.\, A}\ \alpha,$$

or in sequent style:

$$\mathrm{Ind}\ \frac{\Gamma \vdash A[x := \mathbf{0}] \qquad \Gamma, \alpha : A \vdash A[x := \mathsf{succ}(x)]}{\Gamma \vdash \forall x.\, A}\ \alpha.$$

An related axiom is *complete* or *course-of-values induction*, which states that if $A(x)$ holds whenever $A(y)$ holds for all $y < x$ then A holds for any natural number. The axiom for complete induction is written as:

$$(\forall y < x.\, A[x := y]) \to \forall x.\, A$$

While this axiom appear to be stronger than the induction axiom we just defined, it is actually equivalent. This can be seen by considering the standard induction rule for the formula

$$B \equiv \forall y < x.\, A[x := y].$$

For more details see [21, p. 213].

We can write a rule for complete induction in Gentzen's style:

$$\mathrm{CInd}\ \frac{\begin{array}{c}[\forall y < x.\, A[x := y]]^{\alpha}\\ \vdots \\ A\end{array}}{\forall x.\, A}\ \alpha\ ,$$

or in sequent style:

$$\text{CInd}\ \frac{\Gamma, \alpha : \forall y < x.\ A[x := y] \vdash A}{\Gamma \vdash \forall x.\ A}\ \alpha.$$

Now we have all the ingredients we need in order to define Heyting Arithmetic, the standard intuitionistic theory of arithmetic.

Definition 1 (Heyting Arithmetic)**.** *Heyting Arithmetic (*HA*) is defined as the first-order logic theory whose language is the language of arithmetic and whose axioms and rules are the following:*

- *the identity rule,*
- *the rules of minimal logic,*
- *the rule for ex falso quodlibet restricted to atomic formulas,*
- *the axiom and rules for equality,*
- *the axiom and rules for successor, addition and multiplication,*
- *any one of the axioms and rules for induction or complete induction.*

2.1.6 Axioms for the Law of the Excluded Middle

In this subsection we introduce a hierarchy of axiom schemes that are restrictions of the law of the *excluded middle*, taken from [1]. We refer to the same work for explanations and proofs of our claims in this subsection.

We define a purely syntactical version of the usual classification for formulas in prenex normal form in arithmetic.

Definition 2 (Syntactical Arithmetical Hierarchy)**.** *We define the following classes of formulas by induction on n:*

- Π^0_0 *and* Σ^0_0 *are the set of the quantifier free formulas,*
- Π^0_{n+1} *is the set of the formulas* $\forall x.\ A$ *where* $A \in \Sigma^0_n$*,*
- Σ^0_{n+1} *is the set of the formulas* $\forall x.\ A$ *where* $A \in \Pi^0_n$*.*

We do not require closure for logical equivalence and thus the definition of Π^0_n and Σ^0_n is purely syntactical. Since the negation of a formula in prenex normal form is not in prenex normal form we define the following negative conversion, which we call the *dual* of a formula:

$$A^{\neg} \equiv \begin{cases} \exists x.\, B^{\neg} & \text{if } A = \forall x.\, B \in \Pi^0_n, \\ \forall x.\, B^{\neg} & \text{if } A = \exists x.\, B \in \Sigma^0_n, \\ \neg A & \text{if } A \in \Pi^0_0. \end{cases}$$

Note that if A is a Π^0_n (resp. Σ^0_n) formula then $A^{\neg}$ is a Σ^0_n (resp. Π^0_n) formula. Moreover $A^{\neg}$ is classically equivalent to $\neg A$ and intuitionistically stronger than $\neg A$, except when A is a Π^0_0 formula, in which case it is equivalent to $\neg A$.

The law of the excluded middle says that any statement is either true or false. More precisely, it says that either a statement is true or its negation is true. This is intuitionistically equivalent to the fact that either a statement is true or its dual is true. We define a sequence of restricted forms of this law.

Definition 3 (Restricted Excluded Middle Axiom Schemas)**.** *For any $n \in \mathrm{Nat}$, we define the limited law of the excluded middle EM_n as the axiom schema:*

$$A^{\neg} \vee A, \tag{EM_n}$$

where A is a Σ^0_n formula. As a limit case we define EM_∞ where A is a Π^0_n formula for any n.

By adding EM_∞ to intuitionistic logic we get classical logic, so by adding EM_∞ to Heyting Arithmetic we get Peano Arithmetic, the theory of classical arithmetic. We can also produce many intermediate logics by adding EM_n to Heyting Arithmetic, which we write as $\mathsf{HA} + \mathsf{EM}_n$. Note that EM_0 is true in Heyting Arithmetic, so $\mathsf{HA} + \mathsf{EM}_0$ is simply HA.

2.2 A Simply Typed λ-Calculus for Realizability

In this section we introduce system T', a simply typed λ-calculus variant of Gödel's system T in which we shall write our realizers. System T' will be more convenient for our purposes in order to get a more straightforward translation of monads and related concepts from category

theory. There are two main differences between our system T' and system T. The first one is that we replace the boolean type with the more general sum (or co-product) type. The second one is that the recursion operator uses complete recursion instead of standard primitive recursion.

We begin by defining the types. We shall use the metavariables X, Y and Z for types. We assume that we have a finite set of atomic types that includes the unit type Unit and the type of natural numbers Nat. Moreover we have three type binary type constructors $\to, \times, +$. In other words, for any types X and Y we have the arrow (or function) type $X \to Y$, the product type $X \times Y$ and the sum (or co-product) type $X + Y$.

We can now define the typed terms of the calculus. We assume that we have a countable set of typed term constants that includes the constructors and the destructors for the unit, natural, product and sum types (listed in Figure 2.6) and a countable set of variables of type X for any type X:

$$x_0 : X, \ldots, x_n : X, \ldots .$$

We use the metavariables x, y, z for terms. Moreover for any two terms $x : X$ and $y : X \to Y$ we have a term $yx : Y$ and for any variable $x : X$ and term $y : Y$ we have a term $\lambda x.\, y : X \to Y$.

In order to avoid a parenthesis overflow, we shall follow the usual conventions for writing terms and types. For terms this means that application and abstraction are respectively left and right-associative and that abstraction binds as many terms as possible on its right; for types it means that $\times$ and $+$ are left-associative and associate more closely than $\to$, which is right-associative. We also omit outer parenthesis. For example:

$$\begin{array}{lcl} X \to Y \to X \times Y \times Z & \text{stands for} & (X \to (Y \to ((X \times Y) \times Z))), \\ \lambda x^X.\lambda y^Y.\lambda z^Z.t_1 t_2 t_3 & \text{stands for} & (\lambda x^X.(\lambda y^Y.(\lambda z^Z.((t_1 t_2) t_3)))). \end{array}$$

We define some reduction relations, that is, binary relations between terms:

$$(\lambda x^X.t)a \to_\beta t[x := a],$$

$$\mathsf{pr}_{\mathrm{L}}^{X,Y}(\mathsf{pair}^{X,Y}\, ab) \to_\times a, \quad \mathsf{case}^{X,Y,Z}(\mathsf{in}_{\mathrm{L}}^{X,Y}\, a) fg \to_+ fa,$$

$$\mathsf{pr}_{\mathrm{R}}^{X,Y}(\mathsf{pair}^{X,Y}\, ab) \to_\times b, \quad \mathsf{case}^{X,Y,Z}(\mathsf{in}_{\mathrm{R}}^{X,Y}\, b) fg \to_+ gb,$$

$$\mathsf{crec}_n^Z\, h\boldsymbol{m} \to_R \begin{cases} h\boldsymbol{m}(\mathsf{crec}_m^Z\, h) & \text{if } m < n \text{ or } n = \infty, \\ \mathsf{dummy}^Z & \text{otherwise,} \end{cases}$$

Figure 2.6: Constructors and destructors

$$
\begin{gathered}
* : \mathrm{Unit}, \\
\mathsf{pair}^{X,Y} : X \to Y \to X \times Y, \\
\mathsf{pr}_{\mathrm{L}}^{X,Y} : X \times Y \to X, \qquad \mathsf{pr}_{\mathrm{R}}^{X,Y} : X \times Y \to Y, \\
\mathsf{in}_{\mathrm{L}}^{X,Y} : X \to X + Y, \qquad \mathsf{in}_{\mathrm{R}}^{X,Y} : Y \to X + Y, \\
\mathsf{case}^{X,Y,Z} : X + Y \to (X \to Z) \to (Y \to Z) \to Z, \\
\mathsf{zero} : \mathrm{Nat}, \qquad \mathsf{succ} : \mathrm{Nat} \to \mathrm{Nat}, \\
\mathsf{crec}_n^Z : (\mathrm{Nat} \to (\mathrm{Nat} \to Z) \to Z) \to \mathrm{Nat} \to Z.
\end{gathered}
$$

where n is a natural number or the symbol ∞. In order we have the constant constructor of type Unit*, the constructor and the two destructors of the product types, the two constructors and the destructor of the sum types and the two constructors and the destructor of the natural type. Most of those are actually "parametric polymorphic" terms, that is, families of constants indexed by the types X, Y and Z.*

where $a : X$, $b : Y$, $c : Z$, $f : X \to Z$, $g : Y \to Z$ and $h : \mathrm{Nat} \to (\mathrm{Nat} \to Z) \to Z$. Note that we use c as a dummy term of type Z[1]. Then we set the reduction relation $\to$ to be the union of $\to_\beta$, $\to_\times$, $\to_+$ and $\to_R$. Also let $\rightsquigarrow$ be the transitive and reflexive closure of $\to$.

We explain the reduction given for crec, since it is not the standard one. The difference is due to the fact that crec is meant to realize complete induction instead of standard induction. In complete induction, the inductive hypothesis holds not only for the immediate predecessor of the value we are considering, but also for all the smaller values.

Similarly, crec allows us to recursively define a function f where the value of $f(\boldsymbol{m})$ depends not only on the value of $f(\boldsymbol{m-1})$ but also on the value of $f(\boldsymbol{l})$, for any $l < m$. Thus, when computing $\mathsf{crec}^Z_n\, h\boldsymbol{m}$, instead of taking the value of $\mathsf{crec}^Z_n\, h(\boldsymbol{m-1})$ as an argument, h takes the whole function $\mathsf{crec}^Z_n\, h$. In order to avoid unbounded recursion, we add a guard n that prevents $\mathsf{crec}^Z_n\, h$ to be computed on arguments greater or equal to n. More precisely $\mathsf{crec}^Z_n\, h\boldsymbol{m}$ only reduces to $h\boldsymbol{m}(\mathsf{crec}^Z_m\, h)$ if $m < n$; thus, even if h requires $\mathsf{crec}^Z_m\, h$ to be computed on many values, the height of the computation trees is bound by m[2]. Naturally, a "good" h should not evaluate $\mathsf{crec}^Z_m\, h$ on values bigger than m, but in any case the guard guarantees termination. The symbol ∞ acts as a dummy guard, which gets replaced with an effective one when $\mathsf{crec}^Z_\infty\, h$ is evaluated the first time.

We shall also need the following equivalence relations between terms:

$$\lambda x^X.t =_\alpha \lambda y^X.t[x := y], \qquad (\alpha\text{-conversion})$$

$$\lambda x^X.tx =_\eta t, \qquad (\eta\text{-conversion})$$

$$\mathsf{pair}^{X,Y}(\mathsf{pr}^{X,Y}_{\mathrm{L}}\, c)(\mathsf{pr}^{X,Y}_{\mathrm{R}}\, c) =_\times c, \qquad (\times\text{-conversion})$$

$$\mathsf{case}^{X,Y,X+Y}\, d\, \mathsf{in}^{X,Y}_{\mathrm{L}}\, \mathsf{in}^{X,Y}_{\mathrm{R}} =_+ d, \qquad (+\text{-conversion})$$

for all terms t, $c : X \times Y$ and $d : X + Y$. Again we set the equivalence $=$ to be the union of $=_\alpha$, $=_\eta$, $=_\times$ and $=_+$.

[1] As long as the base types are inhabited, we can define an arbitrary dummy term dummy^X for any type X:

$$\mathsf{dummy}^{\mathrm{Unit}} \equiv *, \qquad \mathsf{dummy}^{\mathrm{Nat}} \equiv \mathbf{0},$$
$$\mathsf{dummy}^{X\to Y} \equiv \lambda\_^X.\,\mathsf{dummy}^Y, \quad \mathsf{dummy}^{X\times Y} \equiv \mathsf{pair}\,\mathsf{dummy}^X\,\mathsf{dummy}^Y, \quad \mathsf{dummy}^{X+Y} \equiv \mathsf{in}_{\mathrm{L}}\,\mathsf{dummy}^X.$$

[2] Unlike in standard primitive recursion, where the computation always comprises m steps, in course-of-values primitive recursion the computation can actually be shorter if h "skips" values.

It is easy to see that the boolean type and the related terms true, false and ite in system T can be defined in system T'. The reverse is also true, that is, we can define sum types and terms $\mathsf{in}_{\mathrm{L}}, \mathsf{in}_{\mathrm{R}}$ and case in system T. We show how to interpret the boolean type and related constants of system T in our system.

$$\mathrm{Bool} \equiv \mathrm{Unit} + \mathrm{Unit},$$
$$\mathsf{true} \equiv \mathsf{in}_{\mathrm{L}}\, * \qquad \mathsf{false} \equiv \mathsf{in}_{\mathrm{R}}\, *,$$
$$\mathsf{ite}^{X} \equiv \lambda b^{\mathrm{Bool}}.\lambda x^{X}.\lambda x^{X}.\, \mathsf{case}\, b(\lambda\_^{\mathrm{Unit}}.x)(\lambda\_^{\mathrm{Unit}}.x).$$

We assume that X and Y are inhabited, that is, that there exist terms $x_0 : X$ and $y_0 : Y$. Then we can interpret the sum type and related constants in system T:

$$X + Y \equiv \mathrm{Bool} \times (X \times Y),$$
$$\mathsf{in}_{\mathrm{L}}^{X,Y} \equiv \lambda x^{X}.\, \mathsf{pair}\, \mathsf{true}(\mathsf{pair}\, x y_0),$$
$$\mathsf{in}_{\mathrm{R}}^{X,Y} \equiv \lambda y^{Y}.\, \mathsf{pair}\, \mathsf{false}(\mathsf{pair}\, x_0 y),$$
$$\mathsf{case}^{X,Y,Z} \equiv \lambda b^{\mathrm{Bool}\times(X\times Y)}.\lambda f^{X\to Z}.\lambda g^{Y\to Z}.\, \mathsf{ite}(\mathsf{pr}_{\mathrm{L}}\, b)(f(\mathsf{pr}_{\mathrm{L}}(\mathsf{pr}_{\mathrm{R}}\, b)))(g(\mathsf{pr}_{\mathrm{R}}(\mathsf{pr}_{\mathrm{R}}\, b))).$$

System T' shares most of the good properties of Gödel's system T, in particular confluence, strong normalization[3] and a normal form property.

[3]Strong normalization is a consequence of the explicit bound on recursion given by the subscript in the recursion constant.

Chapter 3

A Monadic Framework for Interactive Realizability

In this chapter we give a new presentation of interactive realizability with a more explicit syntax.

Monads can be used to structure functional programs by providing a clean and modular way to include impure features in purely functional languages. We express interactive realizers by means of an abstract framework that applies the monadic approach used in functional programming to modified realizability, in order to obtain more "relaxed" realizability notions that are suitable to classical logic. In particular we use a combination of the state and exception monads in order to capture the learning-from-mistakes nature of interactive realizers at the syntactic level.

3.1 Introduction

As we have already remarked in the preface, the Curry-Howard correspondence was originally discovered for intuitionistic proofs. This is not coincidental: the type systems needed to interpret intuitionistic proofs are usually very simple and natural, as in the case of Heyting Arithmetic and System T (see [10]). While classical proofs can be transformed into intuitionistic ones by means of the double-negation translation and then translated into typed programs, the existence of a direct correspondence was deemed unlikely until Griffin showed

otherwise in [12].

Starting with Griffin's, other interpretations extending the Curry-Howard correspondence to classical logic have been put forward. Griffin uses a "typed Idealized Scheme" with the control construct call/cc, that allows access to the current continuation. In [19], Parigot introduces the $\lambda\mu$-calculus, an extension of lambda calculus with an additional kind of variables for subterms. In [16], Krivine uses lambda calculus with a non-standard semantics, described by an abstract machine that allows the manipulation of "stacks", which can be thought of as execution contexts.

All these different approaches seems to suggest that, in order to interpret classical logic, we need control operators or some syntactical equivalent thereof. This could be generalized in the idea that "impure" computational constructs are needed in order to interpret non-constructive proofs. Monads are a concept from category theory that has been widely used in computer science. In particular, they can be used to structure functional programs that mimic the effects of impure features.

In [18], Moggi advocates the use of monads as a framework to describe and study many different "notions of computation" in the context of categorical semantics of programming languages. A different take on the same idea that actually eschews category theory completely is suggested in [26] by Wadler: the definition of monad becomes purely syntactic and is used as a framework to structure functional programs by providing a clean and modular way to include impure features in purely functional languages (one noteworthy example is I/O in Haskell).

The main idea of this work is to use monads as suggested by Wadler in order to structure programs extracted from classical proofs by interactive realizability. A program extracted by means interactive realizability, called interactive realizer, can be thought of as a learning process. It accumulate information in a knowledge state and use this knowledge in order to "decide" the instances of EM_1 used in the proof. Since these instances are in general undecidable, the realizer actually makes an "educated guess" about which side of an EM_1 instance is true by looking at the state. Such guesses can be wrong.

This can become apparent later in the proof, when the guessed side of the EM_1 instance is used to deduce some decidable statement. If this decidable statement turns out to be false, then the guess was wrong and the proof cannot be completed. In this case the realizer cannot produce the evidence required for the final statement and fails. However failure is due to the

fact that we made a wrong guess. We can add this information to the state, so that, using this new state, we will be able to guess the EM_1 instance correctly. At this point we discard the computation that occurred after the wrong guess and we resume from there. This time we guess correctly and can proceed until the end or until we fail again because we guessed incorrectly another EM_1 instance.

There are three "impure" parts in the behavior we described: the dependency on the knowledge state, the possibility of failure to produce the intended result and the backtracking after the failure. In this work we use a monadic approach to describe the first two parts which are peculiar to interactive realizability. We do not describe the third part, which is common also to the other interpretations of classical logic.

This chapter is an account of interactive realizability where interactive realizers are encoded as λ-terms following the monadic approach to structuring functional programms suggested by Wadler. We shall prove that our presentation of interactive realizability is a sound semantics for $\mathsf{HA} + \mathsf{EM}_1$.

3.1.1 Main Results

In our presentation, interactive realizer are written in a simply typed λ-calculus with products, coproducts and natural numbers with course-of-value recursion, extended with some abstract terms to represent states and exceptions. The peculiar features of interactive realizability, namely the dependency on the knowledge state and the possibility of failure, are explicitly computed by the λ-terms encoding the realizers. Thus the computational behavior of interactive realizers is evident at the syntactic level, without the need for special semantics.

While proving the soundness of $\mathsf{HA} + \mathsf{EM}_1$ with respect to our definition of interactive realizability, we observed that the soundness of HA did not require any assumption on the specific monad we chose to structure interactive realizers (while the soundness of EM_1 requires them as expected). Prompted by this discovery, we split the presentation in two parts.

The former is an abstract monadic framework for producing realizability notions where the realizers are written in monadic style. We prove that HA is sound with respect to any realizability semantics defined by the framework, for any monad.

The latter is an application of this abstract framework to interactive realizability. We

define a specific monad which we use to structure interactive realizers and a class of realizability semantics that

our definition of interactive realizability can be generalized to an abstract realizability notion where the monad is a parameter. We call this family of realizability notions *monadic realizability* and we show that all instances of monadic realizability are sound semantics for HA. Our interest is in concrete monadic realizability notions that can realize classical principles beyond HA. Another motivation is that in the proof of the soundness of $\mathsf{HA}+\mathsf{EM}_1$ with respect to interactive realizability semantics, we take advantage of the special properties of the interactive realizability monad just to prove the soundness EM_1, while for the rest of HA we just need some generic properties on the semantics which require no assumption on the monad we are considering.

3.1.2 Related Works

This work builds on the presentation of interactive realizability given in [2] by Aschieri and Berardi. The main contributions with respect to [2] is a more precise description of the computational behavior of interactive realizer. This is explained in more detail at the end of this chaper.

Monads have first been used to describe interactive realizability by Berardi and de'Liguoro in [7] and [4], where interactive realizers for $\mathsf{PRA}+\mathsf{EM}_1$ are given a monadic categorical semantics following Moggi's approach. While our idea of using monad to describe interactive realizability was inspired by [7], our work is mostly unrelated: our use of monads follows Wadler's syntactical approach and we employ a different monad that emphasizes different aspects of interactive realizability.

3.2 Monadic Realizability

This section contains the abstract part of our work. We describe the abstract framework of monadic realizability and show the soundness of HA with respect to the semantics induced by a generic monad.

3.2.1 The Monadic Realizability Semantics

In this subsection we define monadic realizability. We state the properties that a suitable relation must satisfy in order to be called a monadic realizability relation and we show how such a relation induces a (monadic) realizability semantics. Then we describe the proof decoration procedure to extract monadic realizers from proofs in HA. Here we are only concerned with proofs in HA, for a non-trivial example of a monadic realizability notion see interactive realizability in Section 3.3.

We start by introducing a syntactic translation of the concept of monad from category theory. Informally, a monad is an operator $T_{\mathfrak{M}}$ "extending" a type, with a canonical embedding from X to $T_{\mathfrak{M}}(X)$, a canonical way to lift a map from X to $T_{\mathfrak{M}}(Y)$ to a map from $T_{\mathfrak{M}}(X)$ to $T_{\mathfrak{M}}(Y)$, a canonical way of merging an element of $T_{\mathfrak{M}}(X)$ and an element of $T_{\mathfrak{M}}(Y)$ into an element of $T_{\mathfrak{M}}(X \times Y)$. We also requires some equations relating these canonical maps, equations which are often satisfied in the practice of programming.

Definition 4 (Syntactic Monad)**.** *A* syntactic monad $\mathfrak{M}$ *is a tuple* $(T_{\mathfrak{M}}, \mathfrak{unit}_{\mathfrak{M}}, \mathfrak{star}_{\mathfrak{M}}, \mathfrak{merge}_{\mathfrak{M}})$ *where* $T_{\mathfrak{M}}$ *is a type constructor, that is, a map from types to types, and, for any types* X, Y, $\mathfrak{unit}_{\mathfrak{M}}$, $\mathfrak{star}_{\mathfrak{M}}$ *and* $\mathfrak{merge}_{\mathfrak{M}}$ *are families (indexed by* X *and* Y*) of closed terms:*

$$\begin{aligned}
\mathfrak{unit}^{X}_{\mathfrak{M}} &: X \to T_{\mathfrak{M}}X,\\
\mathfrak{star}^{X,Y}_{\mathfrak{M}} &: (X \to T_{\mathfrak{M}}Y) \to (T_{\mathfrak{M}}X \to T_{\mathfrak{M}}Y),\\
\mathfrak{merge}^{X,Y}_{\mathfrak{M}} &: T_{\mathfrak{M}}X \to T_{\mathfrak{M}}Y \to T_{\mathfrak{M}}(X \times Y),
\end{aligned}$$

satisfying the following properties:

$$\mathfrak{star}^{X,X}_{\mathfrak{M}}\, \mathfrak{unit}^{X}_{\mathfrak{M}}\, \mathfrak{x} \rightsquigarrow \mathfrak{x}, \tag{M1}$$

$$\mathfrak{star}^{X,Y}_{\mathfrak{M}}\, f(\mathfrak{unit}^{X}_{\mathfrak{M}}\, x) \rightsquigarrow f x, \tag{M2}$$

$$\mathfrak{merge}^{X,Y}_{\mathfrak{M}}(\mathfrak{unit}^{X}_{\mathfrak{M}}\, x)(\mathfrak{unit}^{X}_{\mathfrak{M}}\, y) \rightsquigarrow \mathfrak{unit}^{X\times Y}_{\mathfrak{M}}(\mathsf{pair}^{X,Y}\, xy), \tag{M3}$$

for any $\mathfrak{x} : T_{\mathfrak{M}}X$*,* $f : X \to T_{\mathfrak{M}}Y$*,* $g : Y \to T_{\mathfrak{M}}Z$*,* $x : X$ *and* $x : Y$*.*

The terms $\mathfrak{unit}_{\mathfrak{M}}$ and $\mathfrak{star}_{\mathfrak{M}}$ and Properties M1 and M2 are a straightforward translation of the definition of Kleisli tripe in category theory, an equivalent way to describe a monad[1].

[1] This part of the definition follows the one given by Wadler in [26], with the difference that we replace the

Term $\mathfrak{merge}_{\mathfrak{M}}$ and Property M3 are connected to the definition of strong monad: $\mathfrak{merge}_{\mathfrak{M}}$ is the syntactical counterpart of the natural transformation ϕ, induced by the tensorial strength of the monad (see [18] for details). While ϕ satisfies several other properties in [18], Property M3 is the only one we need for our treatment.

Example 1. *The simplest example of syntactic monad is the* identity monad $\mathfrak{Id}$*, defined as:*

$$T_{\mathfrak{Id}}X \equiv X, \qquad \mathfrak{unit}_{\mathfrak{Id}}^{X} \equiv \lambda x^{X}.x,$$
$$\mathfrak{star}_{\mathfrak{Id}}^{X,Y} \equiv \lambda f^{X\to Y}.f, \qquad \mathfrak{merge}_{\mathfrak{Id}}^{X,Y} \equiv \mathsf{pair}^{X,Y}.$$

This monad cannot describe any additional computational property besides the value a term reduces to.

Example 2. *A simple but non-trivial example is the exception monad $\mathfrak{Ex}$. It describes computations which may either succeed and yield a (normal) value or fail and yield a description of the failure. Consider the usual predecessor function* $\mathsf{pred} : \mathrm{Nat} \to \mathrm{Nat}$ *on the natural numbers: since zero has no predecessor it is common to define* $\mathsf{pred}\,\mathbf{0}$ *as zero. Instead with $\mathfrak{Ex}$ we could have* $\mathsf{pred}\,\mathbf{0}$ *fail and yield a string*[2] *saying "zero has no predecessor".*

Let Ex *be a new ground type and let* $\mathsf{merge} : \mathrm{Ex} \to \mathrm{Ex} \to \mathrm{Ex}$ *be a new constant term. We think terms of type* Ex *as descriptions of failures and we call them* exceptions. *We think of* merge *as an operation that merges the information of multiple exceptions when there are*

term $\mathfrak{bind}$ with $\mathfrak{star}_{\mathfrak{M}}$, where:

$$\mathfrak{bind}^{X,Y} : T_{\mathfrak{M}}X \to (X \to T_{\mathfrak{M}}Y) \to T_{\mathfrak{M}}Y.$$

Defining $\mathfrak{star}_{\mathfrak{M}}$ and $\mathfrak{bind}$ in terms of each other is straightforward:

$$\mathfrak{bind}^{X,Y} \equiv \lambda \mathfrak{x}^{T_{\mathfrak{M}}X}.\lambda f^{X\to T_{\mathfrak{M}}Y}.\,\mathfrak{star}_{\mathfrak{M}}\,f\mathfrak{x},$$
$$\mathfrak{star}_{\mathfrak{M}}^{X,Y} \equiv \lambda f^{X\to T_{\mathfrak{M}}Y}.\lambda \mathfrak{x}^{T_{\mathfrak{M}}X}.\,\mathfrak{bind}\,\mathfrak{x}f.$$

The term $\mathfrak{star}_{\mathfrak{M}}$ corresponds directly to the operator $\_^{*}$ in the definition of Kleisli triple.

[2]assuming we had strings in our calculus

multiple failures in a computations. Now we can define the syntactic monad $\mathfrak{Ex}$ as:

$$
\begin{aligned}
T_{\mathfrak{Ex}}X &\equiv X + \mathrm{Ex}, \qquad \mathfrak{unit}^{X}_{\mathfrak{Ex}} \equiv \lambda x^{X}.\, \mathsf{in}^{X,\mathrm{Ex}}_{L}\, x,\\
\mathfrak{star}^{X,Y}_{\mathfrak{Ex}} &\equiv \lambda f^{X\to Y+\mathrm{Ex}}.\lambda x^{X}.\, \mathsf{case}^{X,\mathrm{Ex},Y+\mathrm{Ex}}\, x f\, \mathsf{in}^{Y,\mathrm{Ex}}_{R},\\
\mathfrak{merge}^{X,Y}_{\mathfrak{Ex}} &\equiv \lambda \mathfrak{x}^{X+\mathrm{Ex}}.\lambda \mathfrak{y}^{Y+\mathrm{Ex}}.\, \mathsf{case}^{X,\mathrm{Ex},(X\times Y)+\mathrm{Ex}}\, \mathfrak{x}\\
&\quad (\lambda x^{X}.\, \mathsf{case}^{Y,\mathrm{Ex},(X\times Y)+\mathrm{Ex}}\, \mathfrak{y} (\lambda y^{Y}.\, \mathsf{in}^{X\times Y,\mathrm{Ex}}_{L} (\mathsf{pair}^{X,Y}\, xy))\, \mathsf{in}^{X\times Y,\mathrm{Ex}}_{R})\\
&\quad (\lambda e^{\mathrm{Ex}}_{1}.\, \mathsf{case}^{Y,\mathrm{Ex},(X\times Y)+\mathrm{Ex}}\, \mathfrak{y} (\lambda y^{Y}.\, \mathsf{in}^{X\times Y,\mathrm{Ex}}_{R}\, e_1)(\lambda e^{\mathrm{Ex}}_{2}.\, \mathsf{in}^{X\times Y,\mathrm{Ex}}_{R}\, \mathsf{merge}\, e_1 e_2)).
\end{aligned}
$$

We omit the proof that $\mathfrak{Ex}$ is a syntactic monad.

A *realizability relation* is a binary relation between terms and closed formulas. When a term and a formula are in such a relation we shall say that the term *realizes* the formula or that the term is a *realizer* of the formula. The intended meaning is that a realizer of a formula is the computational content of a proof of the formula.

We proceed towards the definition of a family of realizability relations, which we call *monadic realizability relations*. Any monadic realizability relation is given with respect to some monad $\mathfrak{M}$ and determines a particular notion of realizability where realizers have the computational properties described by the monad. In the rest of this section we shall assume that $\mathfrak{M} = (T_{\mathfrak{M}}, \mathfrak{unit}_{\mathfrak{M}}, \mathfrak{star}_{\mathfrak{M}}, \mathfrak{merge}_{\mathfrak{M}})$ denotes any fixed syntactic monad.

We now define the type of the monadic realizers of a formula. The idea is to take the standard definition of the type of intuitionistic realizers of a formula A and to apply $T_{\mathfrak{M}}$ only to the type X of the whole formula A and to the types appearing in X after an arrow, namely the types of consequents C of implication sub-formulas $B \to C$ in A and the types of bodies B of universal quantified sub-formulas $\forall x.\, B$ in A. This is the standard call-by-value way to treat arrow types in a monadic framework explained in [25].

Definition 5 (Types for Monadic Realizers)**.** *We define two mappings $\|\cdot\|_{\mathfrak{M}}$ and $|\cdot|_{\mathfrak{M}}$ from formulas to types by simultaneous recursion. The first is the outer or monadic typing of a formula A:*

$$\|A\|_{\mathfrak{M}} = T_{\mathfrak{M}}|A|_{\mathfrak{M}},$$

and the latter is the inner typing, defined by induction on the structure of A:

$$
\begin{aligned}
|P|_{\mathfrak{M}} &= \text{Unit}, & |B \wedge C|_{\mathfrak{M}} &= |B|_{\mathfrak{M}} \times |C|_{\mathfrak{M}},\\
|B \vee C|_{\mathfrak{M}} &= |B|_{\mathfrak{M}} + |C|_{\mathfrak{M}}, & |\exists x.\, B|_{\mathfrak{M}} &= \text{Nat} \times |B|_{\mathfrak{M}},\\
|B \to C|_{\mathfrak{M}} &= |B|_{\mathfrak{M}} \to ||C||_{\mathfrak{M}}, & |\forall x.\, B|_{\mathfrak{M}} &= \text{Nat} \to ||B||_{\mathfrak{M}},
\end{aligned}
$$

where P is an atomic formula and A and B are any formulas.

Note that, we consider $\bot$ to be atomic and $\neg A$ to be a notation for $A \to \bot$, so the types of their realizers follow from the previous definition.

As we defined two types for each formula A, each formula has two possible realizers, one of type $|A|_{\mathfrak{M}}$ and one of type $||A||_{\mathfrak{M}}$. The former will follow the BHK interpretation like an ordinary intuitionistic realizer while the latter will be able to take advantage of the computational properties given by the syntactic monad $\mathfrak{M}$. A formula (in particular classical principles) may have a realizer of monadic type but no realizer of inner type.

Remark 1. *The definition of* $||\cdot||_{\mathfrak{M}}$ *and* $|\cdot|_{\mathfrak{M}}$ *can be derived from the Curry-Howard correspondence between formulas and types and from a call-by-name monadic translation for types. We define the standard interpretation* $|\cdot|$ *that maps a formula into the type of its realizers:*

$$
\begin{aligned}
|P| &= \text{Unit}, & |A \wedge B| &= |A| \times |B|,\\
|A \vee B| &= |A| + |B|, & |A \to B| &= |A| \to |B|,\\
|\forall x.\, A| &= \text{Nat} \to |A|, & |\exists x.\, A| &= \text{Nat} \times |A|.
\end{aligned}
$$

Next we define a translation $[\![\cdot]\!]_{\mathfrak{M}}$ *that lifts types to their monadic counterparts:*

$$
\begin{aligned}
[\![X_0]\!]_{\mathfrak{M}} &\equiv X_0, & [\![X \to Y]\!]_{\mathfrak{M}} &\equiv [\![X]\!]_{\mathfrak{M}} \to T_{\mathfrak{M}}[\![Y]\!]_{\mathfrak{M}},\\
[\![X \times Y]\!]_{\mathfrak{M}} &\equiv [\![X]\!]_{\mathfrak{M}} \times [\![Y]\!]_{\mathfrak{M}}, & [\![X + Y]\!]_{\mathfrak{M}} &\equiv [\![X]\!]_{\mathfrak{M}} + [\![Y]\!]_{\mathfrak{M}},
\end{aligned}
$$

where X_0 *is a ground type. The first two clauses are taken from [27] and the other ones are a simple extension, based on the idea that products and sums behave like ground types.*

By composition we can define the types for the monadic realizers of a formula:

$$
|A|_{\mathfrak{M}} \equiv [\![|A|]\!]_{\mathfrak{M}}, \qquad ||A||_{\mathfrak{M}} \equiv T_{\mathfrak{M}}|A|_{\mathfrak{M}}.
$$

Expanding the definitions we get :

$$\begin{aligned}
|P|_{\mathfrak{M}} &= \text{Unit},\\
|A \wedge B|_{\mathfrak{M}} &= [\![|A|]\!]_{\mathfrak{M}} \times [\![|B|]\!]_{\mathfrak{M}} = |A|_{\mathfrak{M}} \times |B|_{\mathfrak{M}},\\
|A \vee B|_{\mathfrak{M}} &= [\![|A|]\!]_{\mathfrak{M}} + [\![|B|]\!]_{\mathfrak{M}} = |A|_{\mathfrak{M}} + |B|_{\mathfrak{M}},\\
|A \to B|_{\mathfrak{M}} &= [\![|A|]\!]_{\mathfrak{M}} \to T_{\mathfrak{M}}[\![|B|]\!]_{\mathfrak{M}} = |A|_{\mathfrak{M}} \to T_{\mathfrak{M}}|B|_{\mathfrak{M}},\\
|\forall x.\, A|_{\mathfrak{M}} &= [\![\text{Nat}]\!]_{\mathfrak{M}} \to T_{\mathfrak{M}}[\![|B|]\!]_{\mathfrak{M}} = \text{Nat} \to T_{\mathfrak{M}}|A|_{\mathfrak{M}},\\
|\exists x.\, A|_{\mathfrak{M}} &= [\![\text{Nat}]\!]_{\mathfrak{M}} \times [\![|B|]\!]_{\mathfrak{M}} = \text{Nat} \times |A|_{\mathfrak{M}}.
\end{aligned}$$

This is the same translation we described in Definition 5.

We shall now state the requirements for a realizability relation to be a monadic realizability relation. A realizability relation is to be thought of as the restriction of the realizability semantics to closed formulas, that is, a relation between terms of T' and closed formulas which holds when a term is a realizer of the formula. Since a formula can have realizers of inner and outer type, in the following definition two realizability relations will appear: $\mathrm{R}_{\mathfrak{M}}$ for realizers of inner type, whose definition is modeled after the BHK interpretation and $\mathfrak{R}_{\mathfrak{M}}$ for the realizers of outer type, which takes in consideration the computational properties of the monad $\mathfrak{M}$.

As a typographical convention we shall use the letters r, p and q for terms of type $|A|_{\mathfrak{M}}$. Similarly we shall use $\mathfrak{r}$, $\mathfrak{p}$ and $\mathfrak{q}$ for terms of type $||A||_{\mathfrak{M}}$.

Definition 6 (Monadic Realizability Relation)**.** *Let $\mathfrak{R}_{\mathfrak{M}}$ be a realizability relation between terms of type $||A||_{\mathfrak{M}}$ and closed formulas A. Let $\mathrm{R}_{\mathfrak{M}}$ be another realizability relation between terms of type $|A|_{\mathfrak{M}}$ and closed formulas A, such that*

- *$r \;\mathrm{R}_{\mathfrak{M}}\; P$ iff $r \rightsquigarrow *$ and P is true,*
- *$r \;\mathrm{R}_{\mathfrak{M}}\; B \wedge C$ iff $\mathsf{pr}_L\, r \;\mathrm{R}_{\mathfrak{M}}\; B$ and $\mathsf{pr}_R\, r \;\mathrm{R}_{\mathfrak{M}}\; C$,*
- *$r \;\mathrm{R}_{\mathfrak{M}}\; B \vee C$ iff $r \rightsquigarrow \mathsf{in}_L\, a$ and $a \;\mathrm{R}_{\mathfrak{M}}\; B$ or $r \rightsquigarrow \mathsf{in}_R\, b$ and $b \;\mathrm{R}_{\mathfrak{M}}\; C$,*
- *$r \;\mathrm{R}_{\mathfrak{M}}\; B \to C$ iff $rp \;\mathfrak{R}_{\mathfrak{M}}\; C$ for all $p : |B|_{\mathfrak{M}}$ such that $p \;\mathrm{R}_{\mathfrak{M}}\; B$,*
- *$r \;\mathrm{R}_{\mathfrak{M}}\; \forall x.\, B$ iff $r\boldsymbol{n} \;\mathfrak{R}_{\mathfrak{M}}\; B[x := \boldsymbol{n}]$ for all natural numbers n,*

- $r \mathrm{R}_{\mathfrak{M}} \exists x. B$ *iff* $\mathsf{pr}_R\, r \mathrm{R}_{\mathfrak{M}} B[x := \mathsf{pr}_L\, r]$,

where P *is a closed atomic formula and* B *and* C *are generic formulas. We consider* $\perp$ *a closed atomic formula which is never true (for instance* $0 = 1$*). We shall say that the pair* $(\mathfrak{R}_{\mathfrak{M}}, \mathrm{R}_{\mathfrak{M}})$ *is a* monadic realizability relation *if the following properties are satisfied:*

MR1 *if* $r \mathrm{R}_{\mathfrak{M}} A$ *then* $\mathfrak{unit}_{\mathfrak{M}}\, r \,\mathfrak{R}_{\mathfrak{M}} A$,

MR2 *if* $r \mathrm{R}_{\mathfrak{M}} B \to C$ *then* $\mathfrak{star}_{\mathfrak{M}}\, r\mathfrak{p} \,\mathfrak{R}_{\mathfrak{M}} C$ *for all* $\mathfrak{p} : \|B\|_{\mathfrak{M}}$ *such that* $\mathfrak{p} \,\mathfrak{R}_{\mathfrak{M}} B$,

MR3 *if* $\mathfrak{p} \,\mathfrak{R}_{\mathfrak{M}} B$ *and* $\mathfrak{q} \,\mathfrak{R}_{\mathfrak{M}} C$ *then* $\mathfrak{merge}_{\mathfrak{M}}\, \mathfrak{p}\mathfrak{q} \,\mathfrak{R}_{\mathfrak{M}} B \wedge C$.

We will say that a term r *(resp.* $\mathfrak{r}$*) is an* inner *(resp.* outer *or* monadic*)* realizer *of a formula* A *if* $r : |A|_{\mathfrak{M}}$ *(resp.* $r : \|A\|_{\mathfrak{M}}$*) and* $r \mathrm{R}_{\mathfrak{M}} A$ *(resp.* $\mathfrak{r} \,\mathfrak{R}_{\mathfrak{M}} A$*).*

When defining a concrete monadic realizability relation, it is often convenient to define $\mathfrak{R}_{\mathfrak{M}}$ in terms of $\mathrm{R}_{\mathfrak{M}}$ too, that is, the two relations will be defined by simultaneous recursion in terms of each other.

Note how the properties of the relation $\mathrm{R}_{\mathfrak{M}}$ resemble the clauses the definition of standard modified realizability. The main difference is that in the functional cases, those of implication and universal quantification, $\mathrm{R}_{\mathfrak{M}}$ is not defined in terms of itself but uses $\mathfrak{R}_{\mathfrak{M}}$. This makes apparent our claim that the behavior of inner realizers is closely related to the BHK interpretation.

Property MR1 is a constraint on the relationship between $\mathfrak{R}_{\mathfrak{M}}$ and $\mathrm{R}_{\mathfrak{M}}$. It requires $\mathfrak{unit}_{\mathfrak{M}}$ to transform inner realizers into monadic realizers, which can be thought as the fact that realizers satisfying the BHK interpretation are acceptable monadic realizers. Property MR2 again links $\mathrm{R}_{\mathfrak{M}}$ and $\mathfrak{R}_{\mathfrak{M}}$, this time through $\mathfrak{star}_{\mathfrak{M}}$. It says that, if we have a term that maps inner realizers into monadic realizers, its lifting by means of $\mathfrak{star}_{\mathfrak{M}}$ maps monadic realizers into monadic realizers. Property MR3 is a compatibility condition between $\mathfrak{merge}_{\mathfrak{M}}$ and $\mathfrak{R}_{\mathfrak{M}}$. These conditions are all we shall need in order to show that any monadic realizability relation determines a sound semantics for HA. Later we shall see how particular instances of monadic realizability can produce a sound semantics for more than just HA.

Example 3. *We continue our example with the identity monad* $\mathfrak{Id}$ *by defining a monadic realizability relation. We define* $\mathfrak{R}_{\mathfrak{Id}}$ *and* $\mathrm{R}_{\mathfrak{Id}}$ *by simultaneous recursion, with* $\mathrm{R}_{\mathfrak{Id}}$ *defined in*

terms of $\Re_{\mathfrak{I}d}$ *as in Definition 6 and* $\Re_{\mathfrak{I}d}$ *defined as* $\mathrm{R}_{\mathfrak{I}d}$*, which makes sense since* $||A||_{\mathfrak{I}d} = |A|_{\mathfrak{I}d}$.

We can now define the monadic realizability semantics for a given monadic realizability relation, that is, we say when a realizer validates a sequent where a formula can be open and depend on assumptions. In order to do this we need a notation for a formula in a context, which we call *decorated sequent.* A decorated sequent has the form $\Gamma \Vdash_{\mathfrak{M}} r : A$ where A is a formula, r is a term of type $||A||_{\mathfrak{M}}$ and Γ is the context, namely, a list of assumptions written as $\alpha_1 : A_1, \ldots \alpha_k : A_k$ where $A_1, \ldots, A_k$ are formulas and $\alpha_1, \ldots, \alpha_k$ are proof variables that label each assumption, that is, they are variables of type $|A_1|_{\mathfrak{M}}, \ldots, |A_k|_{\mathfrak{M}}$. As we did with the syntactic monad $\mathfrak{M}$, in the following we shall assume to be working with a fixed generic monadic realizability relation $\Re_{\mathfrak{M}}$.

Definition 7 (Monadic Realizability Semantics)**.** *Consider a decorated sequent:*

$$\alpha_1 : A_1, \ldots, \alpha_k : A_k \Vdash_{\mathfrak{M}} \mathfrak{r} : B,$$

such that the free variables of B *are* $x_1, \ldots, x_l$ *and the free variables of* $\mathfrak{r}$ *are either in* $x_1, \ldots, x_l$ *or in* $\alpha_1, \ldots, \alpha_k$*. We say that the sequent is valid if and only if for all natural numbers* $n_1, \ldots, n_l$ *and for all inner realizers* $p_1 : |A_1|_{\mathfrak{M}}, \ldots, p_k : |A_k|_{\mathfrak{M}}$ *such that*

$$p_1 \; \mathrm{R}_{\mathfrak{M}} \; A_1[x_1 := \boldsymbol{n}_1, \ldots, x_l := \boldsymbol{n}_l] \qquad \ldots \qquad p_k \; \mathrm{R}_{\mathfrak{M}} \; A_k[x_1 := \boldsymbol{n}_1, \ldots, x_l := \boldsymbol{n}_l],$$

we have that

$$\mathfrak{r}[x_1 := \boldsymbol{n}_1, \ldots, x_l := \boldsymbol{n}_l, \alpha_1 := p_1, \ldots, \alpha_k := p_k] \; \Re_{\mathfrak{M}} \; A[x_1 := \boldsymbol{n}_1, \ldots, x_l := \boldsymbol{n}_l].$$

Example 4. *From Definition 7, it follows that the semantics induced by the monadic realizability relation* $\Re_{\mathfrak{I}d}$ *is exactly the standard semantics of modified realizability.*

Now that we have defined our semantics, we can illustrate the method to extract monadic realizers from proofs in HA. Later we shall show how to extend our proof extraction technique to $\mathsf{HA} + \mathsf{EM}_1$. Since proof in HA are constructive, the monadic realizers obtained from them behave much like their counterparts in standard modified realizability and comply with the BHK interpretation. In Section 3.3 we shall show how to extend the proof decoration to

non constructive proofs by exhibiting a monadic realizer of EM_1 that truly takes advantage of monadic realizability since it does not act accordingly to the BHK interpretation.

In order to build monadic realizers of proofs in HA we need a generalization of $\mathfrak{star}_{\mathfrak{M}}$ that works for functions of more than one argument. We can build it using $\mathfrak{merge}_{\mathfrak{M}}$ to pack realizers together. Thus let

$$\mathfrak{star}_k^{X_1,\dots,X_k,Y} : (X_1 \to \cdots \to X_k \to T_{\mathfrak{M}}Y) \to (T_{\mathfrak{M}}X_1 \to \cdots \to T_{\mathfrak{M}}X_k \to T_{\mathfrak{M}}Y),$$

be a family of terms defined by induction on $k \geq 0$:

$$\mathfrak{star}_0^Y \equiv \lambda f^{T_{\mathfrak{M}}Y}.f, \qquad \mathfrak{star}_1^{X,Y} \equiv \mathfrak{star}_{\mathfrak{M}}^{X,Y},$$
$$\mathfrak{star}_{k+2} \equiv \lambda f^{X_1\to\cdots\to X_{k+2}\to T_{\mathfrak{M}}Y}.\lambda x^{T_{\mathfrak{M}}X_1}.\lambda y^{T_{\mathfrak{M}}X_2}.\ \mathfrak{star}_{k+1}(\lambda z^{X_1\times X_2}.f(\mathsf{pr_L}\ z)(\mathsf{pr_R}\ z))(\mathfrak{merge}_{\mathfrak{M}}\ xy).$$

For instance:

$$\mathfrak{star}_2 \equiv \lambda f^{X\to Y\to T_{\mathfrak{M}}Z}.\lambda x^{T_{\mathfrak{M}}X}.\lambda y^{T_{\mathfrak{M}}Y}.\ \mathfrak{star}_{\mathfrak{M}}(\lambda z^{X\times Y}.f(\mathsf{pr_L}\ z)(\mathsf{pr_R}\ z))(\mathfrak{merge}_{\mathfrak{M}}\ xy)$$

Moreover we shall need to "raise" the return value of a term $f : X_1 \to \cdots \to X_k \to Y$ with $\mathfrak{unit}_{\mathfrak{M}}$ before we apply $\mathfrak{star}_k$. We define the family of terms $\mathfrak{raise}_k$ by means of $\mathfrak{star}_k$, for any $k \geq 0$:

$$\mathfrak{raise}_k : (X_1 \to \cdots \to X_k \to Y) \to (T_{\mathfrak{M}}X_1 \to \cdots \to T_{\mathfrak{M}}X_k \to T_{\mathfrak{M}}Y)$$
$$\mathfrak{raise}_k \equiv \lambda f^{X_1\to\cdots\to X_k\to Z}.\ \mathfrak{star}_k(\lambda x_1^{X_1}.\cdots\lambda x_k^{X_k}.\ \mathfrak{unit}_{\mathfrak{M}}(f x_1 \cdots x_k)),$$

Now we can show how to extract a monadic realizer from a proof in HA. Let $\mathcal{D}$ be a derivation of some formula A in HA, that is, a derivation ending with $\Gamma \vdash A$. We produce a decorated derivation by replacing each rule instance in $\mathcal{D}$ with the suitable instance of the decorated version of the same rule given in Figure 3.1. These decorated rules differ from the previous version in that they replace sequents with decorated sequents, that is, they bind a term to each formula, where the term bound to the conclusion of a rule is build from the terms bound to the premises. Thus we have defined a term by structural induction on the derivation: if the conclusion of the decorated derivation is $\Gamma \Vdash_{\mathfrak{M}} \mathfrak{r} : A$ then we set $\mathcal{D}^* \equiv \mathfrak{r}$.

In Figure 3.1, the rule labeled Atm shows how to decorate any atomic rule of HA. By definition unfolding, we may check that an atomic rule is interpreted as a kind of "merging" of the information associated to each premise. The nature of the merging depends on the monad we choose.

Figure 3.1: HA rules, decorated with monadic realizers.

$$\text{Id}\;\dfrac{}{\Gamma \Vdash_{\mathfrak{M}} \mathfrak{raise}_0\, x : A} \qquad \text{Atm}\;\dfrac{\Gamma \Vdash_{\mathfrak{M}} \mathfrak{r}_1 : P_1 \qquad \dots \qquad \Gamma \Vdash_{\mathfrak{M}} \mathfrak{r}_l : P_l}{\Gamma \Vdash_{\mathfrak{M}} \mathfrak{raise}_l(\lambda\gamma_1^{\mathrm{Unit}}.\cdots\lambda\gamma_l^{\mathrm{Unit}}.*)\mathfrak{r}_1\cdots\mathfrak{r}_l : P}$$

$$\wedge\text{I}\;\dfrac{\Gamma \Vdash_{\mathfrak{M}} \mathfrak{r}_1 : A \qquad \Gamma \Vdash_{\mathfrak{M}} \mathfrak{r}_2 : B}{\Gamma \Vdash_{\mathfrak{M}} \mathfrak{raise}_2\, \mathsf{pair}\, \mathfrak{r}_1\mathfrak{r}_2 : A \wedge B}$$

$$\wedge\text{E}_\text{L}\;\dfrac{\Gamma \Vdash_{\mathfrak{M}} \mathfrak{r} : A \wedge B}{\Gamma \Vdash_{\mathfrak{M}} \mathfrak{raise}_1\, \mathsf{pr}_\mathrm{L}\, \mathfrak{r} : A} \qquad \wedge\text{E}_\text{R}\;\dfrac{\Gamma \Vdash_{\mathfrak{M}} \mathfrak{r} : A \wedge B}{\Gamma \Vdash_{\mathfrak{M}} \mathfrak{raise}_1\, \mathsf{pr}_\mathrm{R}\, \mathfrak{r} : B}$$

$$\vee\text{I}_\text{R}\;\dfrac{\Gamma \Vdash_{\mathfrak{M}} \mathfrak{r}_1 : A}{\Gamma \Vdash_{\mathfrak{M}} \mathfrak{raise}_1\, \mathsf{in}_\mathrm{L}\, \mathfrak{r}_1 : A \vee B} \qquad \vee\text{I}_\text{L}\;\dfrac{\Gamma \Vdash_{\mathfrak{M}} \mathfrak{r}_2 : B}{\Gamma \Vdash_{\mathfrak{M}} \mathfrak{raise}_1\, \mathsf{in}_\mathrm{R}\, \mathfrak{r}_2 : A \vee B}$$

$$\vee\text{E}\;\dfrac{\Gamma \Vdash_{\mathfrak{M}} \mathfrak{r} : A \vee B \qquad \Gamma, \alpha_{k+1} : A \Vdash_{\mathfrak{M}} \mathfrak{p} : C \qquad \Gamma, \alpha_{k+1} : B \Vdash_{\mathfrak{M}} \mathfrak{q} : C}{\Gamma \Vdash_{\mathfrak{M}} \mathfrak{star}_1(\lambda\gamma^{|A|_{\mathfrak{M}}+|B|_{\mathfrak{M}}}.\, \mathsf{case}\, \gamma(\lambda\alpha_{k+1}^{|A|_{\mathfrak{M}}}.\mathfrak{p})(\lambda\alpha_{k+1}^{|B|_{\mathfrak{M}}}.\mathfrak{q}))\mathfrak{r} : C}$$

$$\to\text{I}\;\dfrac{\Gamma, \alpha_{k+1} : A \Vdash_{\mathfrak{M}} \mathfrak{r} : B}{\Gamma \Vdash_{\mathfrak{M}} \mathfrak{raise}_0(\lambda\alpha_{k+1}^{|A|_{\mathfrak{M}}}.\mathfrak{r}) : A \to B} \qquad \to\text{E}\;\dfrac{\Gamma \Vdash_{\mathfrak{M}} \mathfrak{r} : A \to B \qquad \Gamma \Vdash_{\mathfrak{M}} \mathfrak{p} : A}{\Gamma \Vdash_{\mathfrak{M}} \mathfrak{star}_2(\lambda\gamma_1^{|A|_{\mathfrak{M}}\to|B|_{\mathfrak{M}}}.\lambda\gamma_2^{|A|_{\mathfrak{M}}}.\gamma_1\gamma_2)\mathfrak{r}\mathfrak{p} : B}$$

$$\forall\text{I}\;\dfrac{\Gamma \Vdash_{\mathfrak{M}} \mathfrak{r} : A}{\Gamma \Vdash_{\mathfrak{M}} \mathfrak{raise}_0(\lambda x^{\mathrm{Nat}}.\mathfrak{r}) : \forall x.\, A} \qquad \forall\text{E}\;\dfrac{\Gamma \Vdash_{\mathfrak{M}} \mathfrak{r} : \forall x.\, A}{\Gamma \Vdash_{\mathfrak{M}} (\mathfrak{star}_1(\lambda\gamma^{\mathrm{Nat}\to\|A\|_{\mathfrak{M}}}.\gamma t))\mathfrak{r} : A[x := t]}$$

$$\exists\text{I}\;\dfrac{\Gamma \Vdash_{\mathfrak{M}} \mathfrak{r} : A[x := t]}{\Gamma \Vdash_{\mathfrak{M}} \mathfrak{raise}_1(\lambda\gamma^{|A|_{\mathfrak{M}}}.\, \mathsf{pair}\, t\gamma)\mathfrak{r} : \exists x.\, A}$$

$$\exists\text{E}\;\dfrac{\Gamma \Vdash_{\mathfrak{M}} r_1 : \exists x.\, A \qquad \Gamma, \alpha : A[x := y] \Vdash_{\mathfrak{M}} r_2 : C}{\Gamma \Vdash_{\mathfrak{M}} \mathfrak{star}_1(\lambda\gamma^{\mathrm{Nat}\times|A|_{\mathfrak{M}}}.(\lambda y^{\mathrm{Nat}}.\lambda\alpha^{|A|_{\mathfrak{M}}}.r_2)(\mathsf{pr}_\mathrm{L}\,\gamma)(\mathsf{pr}_\mathrm{R}\,\gamma))r_1 : C}$$

$$\text{Ind}\;\dfrac{\Gamma, \alpha_{k+1} : \forall z.\, z < y \to A[x := z] \Vdash_{\mathfrak{M}} r : A[x := y]}{\Gamma \Vdash_{\mathfrak{M}} \mathfrak{raise}_0(\mathsf{crec}_\infty\, f) : \forall x.\, A}$$

where all formulas in rule Atm are atomic, t is any term and f is defined as follows:

$$f \equiv \lambda y^{\mathrm{Nat}}.\lambda\beta^{\mathrm{Nat}\to T_{\mathfrak{M}}|A|_{\mathfrak{M}}}.(\lambda\alpha^{\mathrm{Nat}\to T_{\mathfrak{M}}(\mathrm{Unit}\to T_{\mathfrak{M}}|A|_{\mathfrak{M}})}.r)(\lambda z^{\mathrm{Nat}}.\, \mathfrak{raise}_0(\lambda\_^{\mathrm{Unit}}.\beta z)),$$

with β not free in r.

Remark 2. *In Figure 3.1, we wrote all realizers using only $\mathfrak{raise}_k$ and $\mathfrak{star}_k$ for the sake of consistency, but note that $\mathfrak{raise}_0$ could have been replaced by $\mathfrak{unit}_{\mathfrak{M}}$ since it reduces to it:*

$$\begin{aligned} \mathfrak{raise}_0 &\equiv_{\mathfrak{raise}_0} \lambda f^Z.\, \mathfrak{star}_0(\mathfrak{unit}_{\mathfrak{M}}\, f) \\ &\equiv_{\mathfrak{star}_0} \lambda f^Z.(\lambda f^{T_{\mathfrak{M}}Z}.f)(\mathfrak{unit}_{\mathfrak{M}}\, f) \\ &\to_\beta \lambda f^Z.\, \mathfrak{unit}_{\mathfrak{M}}\, f \end{aligned}$$

$$=_\eta \mathfrak{unit}_{\mathfrak{M}}$$

Moreover $\mathfrak{raise}_2$ *pair reduces to* $\mathfrak{merge}_{\mathfrak{M}}$*:*

$$
\begin{aligned}
\mathfrak{raise}_2\, \mathsf{pair} &\equiv_{\mathfrak{raise}_2} (\lambda f^{X\to Y\to X\times Y}.\, \mathfrak{star}_2(\lambda x^X.\lambda y^Y.\, \mathfrak{unit}_{\mathfrak{M}}(f x y)))\, \mathsf{pair} \\
&\rightsquigarrow_\beta \mathfrak{star}_2(\lambda x^X.\lambda y^Y.\, \mathfrak{unit}_{\mathfrak{M}}(\mathsf{pair}\, x y)) \\
&\equiv_{\mathfrak{star}_2} (\lambda f^{X\to Y\to T_{\mathfrak{M}}(X\times Y)}.\lambda x^{T_{\mathfrak{M}}X}.\lambda y^{T_{\mathfrak{M}}Y}. \\
&\qquad \mathfrak{star}_k(\lambda z^{X\times Y}.f(\mathsf{pr}_L\, z)(\mathsf{pr}_R\, z))(\mathfrak{merge}_{\mathfrak{M}}\, xy))(\lambda x^X.\lambda y^Y.\, \mathfrak{unit}_{\mathfrak{M}}\, \mathsf{pair}\, xy) \\
&\rightsquigarrow_\beta \lambda x^{T_{\mathfrak{M}}X}.\lambda y^{T_{\mathfrak{M}}Y}.\, \mathfrak{star}_{\mathfrak{M}}(\lambda z^{X\times Y}.(\lambda x^X.\lambda y^Y.\, \mathfrak{unit}_{\mathfrak{M}}\, \mathsf{pair}\, xy)(\mathsf{pr}_L\, z)(\mathsf{pr}_R\, z))(\mathfrak{merge}_{\mathfrak{M}}\, xy) \\
&\rightsquigarrow_\beta \lambda x^{T_{\mathfrak{M}}X}.\lambda y^{T_{\mathfrak{M}}Y}.\, \mathfrak{star}_{\mathfrak{M}}(\lambda z^{X\times Y}.\, \mathfrak{unit}_{\mathfrak{M}}\, \mathsf{pair}\, \mathsf{pr}_L\, z \mathsf{pr}_R\, z)(\mathfrak{merge}_{\mathfrak{M}}\, xy) \\
&=_\times \lambda x^{T_{\mathfrak{M}}X}.\lambda y^{T_{\mathfrak{M}}Y}.\, \mathfrak{star}_{\mathfrak{M}}(\lambda z^{X\times Y}.\, \mathfrak{unit}_{\mathfrak{M}}\, z)(\mathfrak{merge}_{\mathfrak{M}}\, xy) \\
&=_\eta \lambda x^{T_{\mathfrak{M}}X}.\lambda y^{T_{\mathfrak{M}}Y}.\, \mathfrak{star}_{\mathfrak{M}}\, \mathfrak{unit}_{\mathfrak{M}}(\mathfrak{merge}_{\mathfrak{M}}\, xy) \\
&\rightsquigarrow_{M2} \lambda x^{T_{\mathfrak{M}}X}.\lambda y^{T_{\mathfrak{M}}Y}.\, \mathfrak{merge}_{\mathfrak{M}}\, xy \\
&=_\eta \mathfrak{merge}_{\mathfrak{M}},
\end{aligned}
$$

so we could replace it in $\wedge$I.

Note how the monadic realizer of each rule is obtained by lifting the suitable term in the corresponding standard modified realizer with $\mathfrak{star}_k$ or $\mathfrak{raise}_k$. These monadic realizers do not take advantages of particular monadic features (it cannot be otherwise since we have made no assumption on the syntactic monad or the monadic realizability relation). The main difference is that they can act as "glue" between "true" monadic realizers of non constructive axioms and rules, for instance the one we shall build in Section 3.3.

Here we can see that monadic realizability generalizes intuitionistic realizability: decorated rules in Figure 3.1 reduce to the standard decorated rules for intuitionistic modified realizability in the case of the identity monad $\mathfrak{I}d$.

3.2.2 The Soundness Theorem

In this subsection we prove that HA is sound with respect to the monadic realizability semantics given in Definition 7. This amounts to say that we can use proof decoration to extract, from any proof in HA, a monadic realizer that makes its conclusion valid. We prove this for a

generic monad, which means that the soundness of HA does not depend on the special properties of any specific monad. The proof only needs the simple properties we have requested in Definition 6.

Before proving the main result, we need to show that $\mathfrak{star}_k$ and $\mathfrak{raise}_k$ satisfy a generalization of Property MR2.

Proposition 1 (Monadic Realizability Property for $\mathfrak{star}_k$)**.** *Let $A_1, \ldots, A_k$ and B be any formulas and let $r : |A_1|_{\mathfrak{M}} \to \cdots \to |A_k|_{\mathfrak{M}} \to \|B\|_{\mathfrak{M}}$ be a term. Assume that, for all terms $p_1 : |A_1|_{\mathfrak{M}}, \ldots, p_k : |A_k|_{\mathfrak{M}}$ such that $p_1 \ \mathrm{R}_{\mathfrak{M}}\ A_1, \ldots, p_k \ \mathrm{R}_{\mathfrak{M}}\ A_k$, we have:*

$$r p_1 \cdots p_k \ \mathfrak{R}_{\mathfrak{M}}\ B.$$

Then, for all terms $\mathfrak{p}_1 : \|A_1\|_{\mathfrak{M}}, \ldots, \mathfrak{p}_k : \|A_k\|_{\mathfrak{M}}$ such that $\mathfrak{p}_1 \ \mathfrak{R}_{\mathfrak{M}}\ A_1, \ldots, \mathfrak{p}_k \ \mathfrak{R}_{\mathfrak{M}}\ A_k$, we have:

$$\mathfrak{star}_k\, r \mathfrak{p}_1 \cdots \mathfrak{p}_k \ \mathfrak{R}_{\mathfrak{M}}\ B.$$

Proof. By induction on k. For $k = 0$ it is trivial and for $k = 1$ it follows from Property MR2 since $\mathfrak{star}_1 \equiv \mathfrak{star}_{\mathfrak{M}}$. Now we just need to prove that if the statement holds for some $k \geq 1$, it holds for $k + 1$ too.

As in the statement we assume that, for all terms $p_1 : |A_1|_{\mathfrak{M}}, \ldots, p_{k+1} : |A_{k+1}|_{\mathfrak{M}}$ such that $p_1 \ \mathrm{R}_{\mathfrak{M}}\ A_1, \ldots, p_{k+1} \ \mathrm{R}_{\mathfrak{M}}\ A_{k+1}$:

$$r p_1 \cdots p_{k+1} \ \mathfrak{R}_{\mathfrak{M}}\ B,$$

and that $\mathfrak{p}_1 : \|A_1\|_{\mathfrak{M}}, \ldots, \mathfrak{p}_{k+1} : \|A_{k+1}\|_{\mathfrak{M}}$ are terms such that $\mathfrak{p}_1 \ \mathfrak{R}_{\mathfrak{M}}\ A_1, \ldots, \mathfrak{p}_{k+1} \ \mathfrak{R}_{\mathfrak{M}}\ A_{k+1}$. We need to show that:

$$\mathfrak{star}_{k+1}\, r \mathfrak{p}_1 \cdots \mathfrak{p}_{k+1} \ \mathfrak{R}_{\mathfrak{M}}\ B.$$

Since we know by definition of $\mathfrak{star}_{k+1}$ that $\mathfrak{star}_{k+1}\, r \mathfrak{p}_1 \cdots \mathfrak{p}_{k+1}$ reduces to the term:

$$\mathfrak{star}_k (\lambda z^{|A_1|_{\mathfrak{M}} \times |A_2|_{\mathfrak{M}}}.r(\mathsf{pr}_{\mathrm{L}}\, z)(\mathsf{pr}_{\mathrm{R}}\, z))(\mathfrak{merge}_{\mathfrak{M}}\, \mathfrak{p}_1 \mathfrak{p}_2)\mathfrak{p}_3 \cdots \mathfrak{p}_{k+1},$$

and by Property MR3 that $\mathfrak{merge}_{\mathfrak{M}}\, \mathfrak{p}_1 \mathfrak{p}_2 \ \mathfrak{R}_{\mathfrak{M}}\ A_1 \wedge A_2$, we see that we can use the inductive hypothesis on k to conclude. In order to do so we have to show that the assumption of the inductive hypothesis holds, namely that, for any $p_1 : |A_1|_{\mathfrak{M}} \times |A_2|_{\mathfrak{M}}, p_3 : |A_3|_{\mathfrak{M}}, \ldots, p_k : |A_k|_{\mathfrak{M}}$ such that $p_1 \ \mathrm{R}_{\mathfrak{M}}\ A_1 \wedge A_2$, $p_2 \ \mathrm{R}_{\mathfrak{M}}\ A_2, \ldots, p_k \ \mathrm{R}_{\mathfrak{M}}\ A_k$ it is the case that:

$$(\lambda z^{|A_1|_{\mathfrak{M}} \times |A_2|_{\mathfrak{M}}}.r(\mathsf{pr}_{\mathrm{L}}\, z)(\mathsf{pr}_{\mathrm{R}}\, z)) p_1 \cdots p_k \ \mathfrak{R}_{\mathfrak{M}}\ B.$$

By reducing the realizer we get that this is equivalent to:

$$r(\mathsf{pr}_{\mathrm{L}}\, p_1)(\mathsf{pr}_{\mathrm{R}}\, p_1)p_2 \cdots p_k \ \mathfrak{R}_{\mathfrak{M}}\ B,$$

which is true by the assumption on r since $p_1 \ \mathrm{R}_{\mathfrak{M}}\ A_1 \wedge A_2$ means that $\mathsf{pr}_{\mathrm{L}}\, p_1 \ \mathrm{R}_{\mathfrak{M}}\ A_1$ and $\mathsf{pr}_{\mathrm{R}}\, p_1 \ \mathrm{R}_{\mathfrak{M}}\ A_2$ by definition of $\mathrm{R}_{\mathfrak{M}}$. □

We prove a similar property for $\mathfrak{raise}_k$.

Proposition 2 (Monadic Realizability Property for $\mathfrak{raise}_k$)**.** *Let $A_1, \ldots, A_k$ and B be any formulas and let $r : |A_1|_{\mathfrak{M}} \to \cdots \to |A_k|_{\mathfrak{M}} \to |B|_{\mathfrak{M}}$ be a term. Assume that, for all terms $p_1 : |A_1|_{\mathfrak{M}}, \ldots, p_k : |A_k|_{\mathfrak{M}}$ such that $p_1 \ \mathrm{R}_{\mathfrak{M}}\ A_1, \ldots, p_k \ \mathrm{R}_{\mathfrak{M}}\ A_k$, it is the case that:*

$$r p_1 \cdots p_k \ \mathrm{R}_{\mathfrak{M}}\ B.$$

Then, for all terms $\mathfrak{p}_1 : ||A_1||_{\mathfrak{M}}, \ldots, \mathfrak{p}_k : ||A_k||_{\mathfrak{M}}$ such that $\mathfrak{p}_1 \ \mathfrak{R}_{\mathfrak{M}}\ A_1, \ldots, \mathfrak{p}_k \ \mathfrak{R}_{\mathfrak{M}}\ A_k$, we have that:

$$\mathfrak{raise}_k\, r \mathfrak{p}_1 \cdots \mathfrak{p}_k \ \mathfrak{R}_{\mathfrak{M}}\ B.$$

Proof. Assume that, for all terms $p_1 : |A_1|_{\mathfrak{M}}, \ldots, p_k : |A_k|_{\mathfrak{M}}$ such that $p_1 \ \mathrm{R}_{\mathfrak{M}}\ A_1, \ldots, p_k \ \mathrm{R}_{\mathfrak{M}}\ A_k$, it is the case that:

$$r p_1 \cdots p_k \ \mathrm{R}_{\mathfrak{M}}\ B,$$

and let $\mathfrak{p}_1 : ||A_1||_{\mathfrak{M}}, \ldots, \mathfrak{p}_k : ||A_k||_{\mathfrak{M}}$ be terms such that $\mathfrak{p}_1 \ \mathfrak{R}_{\mathfrak{M}}\ A_1, \ldots, \mathfrak{p}_k \ \mathfrak{R}_{\mathfrak{M}}\ A_k$. We want to prove that:

$$\mathfrak{raise}_k\, r \mathfrak{p}_1 \cdots \mathfrak{p}_k \ \mathfrak{R}_{\mathfrak{M}}\ B.$$

By definition of $\mathfrak{raise}_k$ this reduces to:

$$\mathfrak{star}_k(\lambda x_1^{|A_1|_{\mathfrak{M}}}. \cdots \lambda x_k^{|A_k|_{\mathfrak{M}}}. \, \mathfrak{unit}_{\mathfrak{M}}(r x_1 \cdots x_k))\mathfrak{p}_1 \cdots \mathfrak{p}_k \ \mathfrak{R}_{\mathfrak{M}}\ B.$$

This follows by Proposition 1 if we can show that, for any $p_1 : |A_1|_{\mathfrak{M}}, \ldots, p_k : |A_k|_{\mathfrak{M}}$ such that $p_1 \ \mathrm{R}_{\mathfrak{M}}\ A_1, \ldots, p_k \ \mathrm{R}_{\mathfrak{M}}\ A_k$, we have:

$$(\lambda x_1^{|A_1|_{\mathfrak{M}}}. \cdots \lambda x_k^{|A_k|_{\mathfrak{M}}}. \, \mathfrak{unit}_{\mathfrak{M}}(r x_1 \cdots x_k))p_1 \cdots p_k \ \mathfrak{R}_{\mathfrak{M}}\ B.$$

Reducing the realizer we get that this is equivalent to:

$$\mathfrak{unit}_{\mathfrak{M}}(r p_1 \cdots p_k) \ \mathfrak{R}_{\mathfrak{M}}\ B,$$

and this follows by Property MR1 and by assumption on r. □

Now we are ready to prove the soundness theorem.

Theorem 1 (Soundness of HA with respect to the Monadic Realizability Semantics)**.** *Let $\mathcal{D}$ be a derivation of $\Gamma \vdash A$ in HA and $\mathfrak{R}_{\mathfrak{M}}$ a monadic realizability relation. Then $\Gamma \Vdash_{\mathfrak{M}} \mathcal{D}^* : A$ is valid with respect to $\mathfrak{R}_{\mathfrak{M}}$.*

The proof is long but simple, proceeding by induction on the structure of the decorated version of $\mathcal{D}$.

Proof. We proceed by induction on the structure of the decorated version of $\mathcal{D}$, that is, we assume that the statement holds for all decorated sub-derivations of $\mathcal{D}$ and we prove that it holds for $\mathcal{D}$ too. More precisely we have to check the soundness of each decorated rule, showing that the validity of the premises yields the validity of the conclusion.

We start with some general notation and observations. Let $\Gamma \equiv \alpha_1 : A_1, \ldots, \alpha_k : A_k$ for some k. Following the notation in Definition 7, we fix natural numbers $n_1, \ldots, n_l$ and terms $r_1 : A_1, \ldots, r_k : A_k$, we define abbreviations:

$$\Omega \equiv x_1 \coloneqq \boldsymbol{n}_1, \ldots, x_l \coloneqq \boldsymbol{n}_l,$$

$$\Sigma \equiv \alpha_1 \coloneqq r_1, \ldots, \alpha_k \coloneqq r_k,$$

and we assume that:

$$r_1 \ \mathrm{R}_{\mathfrak{M}} \ A_1[\Omega] \qquad \ldots \qquad r_k \ \mathrm{R}_{\mathfrak{M}} \ A_k[\Omega].$$

Note that if some term $t : X_1 \to \cdots \to X_k \to Y$ has no free variables then $(ta_1 \cdots a_k)[\Omega, \Sigma] \equiv t(a_1[\Omega, \Sigma]) \cdots (a_k[\Omega, \Sigma])$. In particular this holds if t is one of $\mathfrak{star}_k$, $\mathfrak{raise}_k$, pair, pr_{L}, pr_{R}, case, in_{L}, in_{R}. The same holds for formulas, so $(A \star B)[\Omega] \equiv A[\Omega] \star B[\Omega]$ where $\star$ is one of $\wedge$, $\vee$ or $\to$. Also note that $|A[\Omega]|_{\mathfrak{M}} = |A|_{\mathfrak{M}}$ since $|\cdot|_{\mathfrak{M}}$ does not depend on the terms in A. In particular the types of the proof variables in Γ do not change, meaning we do not need to perform substitutions in Γ. We shall take advantage of these facts without mentioning it.

Now we can start showing that the rules are sound.

Id We have to prove that:

$$(\mathfrak{raise}_0\, \alpha_i)[\Omega, \Sigma] \ \mathfrak{R}_{\mathfrak{M}} \ A[\Omega],$$

where $A = A_i$ for some $i \in \{1, \ldots, k\}$.

By performing the substitutions, we can rewrite the realizer as $\mathfrak{raise}_0\, r_i$ so we need to prove that:

$$\mathfrak{raise}_0\, r_i \;\Re_{\mathfrak{M}}\; A.$$

This follows by Proposition 2 since by assumption $r_i \;\mathrm{R}_{\mathfrak{M}}\; A_i[\Omega]$.

Atm We have to prove that:

$$(\mathfrak{raise}_l(\lambda\gamma_1^{\mathrm{Unit}}.\cdots\lambda\gamma_l^{\mathrm{Unit}}.*)\mathfrak{r}_1\cdots\mathfrak{r}_l)[\Omega,\Sigma] \;\Re_{\mathfrak{M}}\; P[\Omega].$$

By performing the substitutions, we can rewrite the realizer as:

$$\mathfrak{raise}_l(\lambda\gamma_1^{\mathrm{Unit}}.\cdots\lambda\gamma_l^{\mathrm{Unit}}.*)\mathfrak{r}_1[\Omega,\Sigma]\cdots\mathfrak{r}_l[\Omega,\Sigma].$$

By inductive hypothesis we know that

$$\mathfrak{r}_1[\Omega,\Sigma] \;\Re_{\mathfrak{M}}\; P_1[\Omega],\ldots,\mathfrak{r}_l[\Omega,\Sigma] \;\Re_{\mathfrak{M}}\; P_l[\Omega],$$

and thus we can conclude by Proposition 2 if we can show that:

$$(\lambda\gamma_1^{\mathrm{Unit}}.\cdots\lambda\gamma_l^{\mathrm{Unit}}.*)r_1\cdots r_l \;\mathrm{R}_{\mathfrak{M}}\; P[\Omega],$$

for all $r_1,\ldots,r_l$ that are inner realizers of $P_1,\ldots,P_l$ respectively. Since

$$(\lambda\gamma_1^{\mathrm{Unit}}.\cdots\lambda\gamma_l^{\mathrm{Unit}}.*)r_1\cdots r_l,$$

reduces to $*$ and $* \;\mathrm{R}_{\mathfrak{M}}\; P[\Omega]$ by definition of $\mathrm{R}_{\mathfrak{M}}$ we are done.

In the following we will apply the substitutions directly without mentioning it.

$\wedge$I We have to prove that

$$\mathfrak{raise}_2\ \mathsf{pair}\ \mathfrak{p}[\Omega,\Sigma]\mathfrak{q}[\Omega,\Sigma] \;\Re_{\mathfrak{M}}\; A[\Omega]\wedge B[\Omega],$$

assuming that $\mathfrak{p}[\Omega,\Sigma] \;\Re_{\mathfrak{M}}\; A[\Omega]$ and $\mathfrak{q}[\Omega,\Sigma] \;\Re_{\mathfrak{M}}\; A[\Omega]$. This follows by Proposition 2 since

$$\mathsf{pair}\ pq \;\mathrm{R}_{\mathfrak{M}}\; A\wedge B,$$

for all inner realizers p of A and q of B, by definition of $\mathrm{R}_{\mathfrak{M}}$.

$\wedge E_L$ We have to prove that

$$(\mathfrak{raise}_1\ \mathsf{pr}_L\ \mathfrak{r})[\Omega, \Sigma]\ \mathfrak{R}_{\mathfrak{M}}\ A[\Omega],$$

assuming that

$$\mathfrak{r}[\Omega, \Sigma]\ \mathfrak{R}_{\mathfrak{M}}\ A[\Omega] \wedge B[\Omega].$$

This follows by Proposition 2 if

$$\mathsf{pr}_L\ r\ \mathrm{R}_{\mathfrak{M}}\ A[\Omega],$$

for any inner realizer r of $A[\Omega] \wedge B[\Omega]$. This is the case because from $r\ \mathrm{R}_{\mathfrak{M}}\ A \wedge B$ if and only if $\mathsf{pr}_L\ r\ \mathrm{R}_{\mathfrak{M}}\ A$ by definition of $\mathrm{R}_{\mathfrak{M}}$.

$\wedge E_R$ Very similar to the proof for $\wedge E_L$.

$\vee I_L$ We have to show that:

$$\mathfrak{raise}_1\ \mathsf{in}_L\ \mathfrak{p}[\Sigma, \Omega]\ \mathfrak{R}_{\mathfrak{M}}\ A[\Omega] \vee B[\Omega],$$

assuming that:

$$\mathfrak{p}[\Sigma, \Omega]\ \mathfrak{R}_{\mathfrak{M}}\ A[\Omega].$$

This follows by Proposition 2 if

$$\mathsf{in}_L\ p\ \mathrm{R}_{\mathfrak{M}}\ A[\Omega],$$

for any inner realizer p of $A[\Omega]$. This is the case since $p\ \mathrm{R}_{\mathfrak{M}}\ A[\Omega]$ if and only if $\mathsf{in}_L\ p\ \mathrm{R}_{\mathfrak{M}}\ A[\Omega] \vee B[\Omega]$ by definition of $\mathrm{R}_{\mathfrak{M}}$.

$\vee I_R$ Very similar to the proof for $\vee I_L$.

$\vee E$ We have to show that:

$$\mathfrak{star}_1(\lambda\gamma^{|A|_{\mathfrak{M}}+|B|_{\mathfrak{M}}}.\,\mathsf{case}\,\gamma(\lambda\alpha^{|A|_{\mathfrak{M}}}.\mathfrak{p}[\Omega, \Sigma])(\lambda\beta^{|B|_{\mathfrak{M}}}.\mathfrak{q}[\Omega, \Sigma]))\mathfrak{r}[\Omega, \Sigma]\ \mathfrak{R}_{\mathfrak{M}}\ C[\Omega]$$

assuming by inductive hypothesis that:

1. $\mathfrak{r}[\Omega, \Sigma]\ \mathfrak{R}_{\mathfrak{M}}\ A[\Omega] \vee B[\Omega]$,
2. $\mathfrak{p}[\Omega, \Sigma, \alpha := p]\ \ \mathfrak{R}_{\mathfrak{M}}\ C[\Omega]$ for any inner realizer p of $A[\Omega]$,

3. $\mathfrak{q}[\Omega, \Sigma, \beta := q] \; \Re_{\mathfrak{M}} \; C[\Omega]$ for any inner realizer q of $B[\Omega]$.

We can conclude by Proposition 1 if we show that

$$(\lambda\gamma^{|A|_{\mathfrak{M}}+|B|_{\mathfrak{M}}}.\,\mathsf{case}\,\gamma(\lambda\alpha^{|A|_{\mathfrak{M}}}.\mathfrak{p}[\Omega,\Sigma])(\lambda\beta^{|B|_{\mathfrak{M}}}.\mathfrak{q}[\Omega,\Sigma]))r,$$

which β-reduces to

$$\mathsf{case}\,r(\lambda\alpha^{|A|_{\mathfrak{M}}}.\mathfrak{p}[\Omega,\Sigma])(\lambda\beta^{|B|_{\mathfrak{M}}}.\mathfrak{q}[\Omega,\Sigma]), \tag{3.1}$$

is a monadic realizer of $C[\Omega]$ for any inner realizer r of $A[\Omega] \vee B[\Omega]$.

By definition of $\mathrm{R}_{\mathfrak{M}}$, we know that either $r \rightsquigarrow \mathsf{in}_{\mathrm{L}}\, p$ where p is an inner realizer of $A[\Omega]$ or $r \rightsquigarrow \mathsf{in}_{\mathrm{R}}\, q$ where q is an inner realizer of $B[\Omega]$. Assume that we are in the first case (the second case is analogous). Then (3.1) becomes:

$$\mathsf{case}(\mathsf{in}_{\mathrm{L}}\, p)(\lambda\alpha^{|A|_{\mathfrak{M}}}.\mathfrak{p}[\Omega,\Sigma])(\lambda\beta^{|B|_{\mathfrak{M}}}.\mathfrak{q}[\Omega,\Sigma]),$$

which reduces to

$$(\lambda\alpha^{|A|_{\mathfrak{M}}}.\mathfrak{p}[\Omega,\Sigma])p,$$

and to

$$\mathfrak{p}[\Omega,\Sigma,\alpha := p],$$

which is a monadic realizer of $C[\Omega]$ by inductive hypothesis.

$\to$ I We have to show that:

$$\mathfrak{raise}_0(\lambda\alpha_{k+1}^{|A|_{\mathfrak{M}}}.\mathfrak{r}[\Omega,\Sigma]) \; \Re_{\mathfrak{M}} \; A[\Omega] \to B[\Omega],$$

assuming that:

$$\mathfrak{r}[\Omega,\Sigma,\alpha_{k+1} := p] \; \Re_{\mathfrak{M}} \; B[\Omega],$$

for any inner realizer p of $A[\Omega]$. By Proposition 2 it is enough to show that:

$$\lambda\alpha_{k+1}^{|A|_{\mathfrak{M}}}.\mathfrak{r}[\Omega,\Sigma] \; \mathrm{R}_{\mathfrak{M}} \; A[\Omega] \to B[\Omega].$$

By definition of $\mathrm{R}_{\mathfrak{M}}$ this holds if and only if:

$$(\lambda\alpha_{k+1}^{|A|_{\mathfrak{M}}}.\mathfrak{r}[\Omega,\Sigma])p \; \Re_{\mathfrak{M}} \; B[\Omega],$$

for any inner realizer p of $A[\Omega]$. Reducing we get:

$$\mathfrak{r}[\Omega,\Sigma][\alpha_{k+1} := p]) \Re_{\mathfrak{M}} B[\Omega],$$

and since $\mathfrak{r}[\Omega,\Sigma][\alpha_{k+1} := p] \equiv \mathfrak{r}[\Omega,\Sigma,\alpha_{k+1} := p]$, we can conclude by the inductive hypothesis.

$\to$ E We have to show that:

$$(\mathfrak{star}_2(\lambda\gamma_1^{|A|_{\mathfrak{M}}\to|B|_{\mathfrak{M}}}.\lambda\gamma_2^{|A|_{\mathfrak{M}}}.\gamma_1\gamma_2)\mathfrak{r}[\Omega,\Sigma]\mathfrak{p}[\Omega,\Sigma]) \Re_{\mathfrak{M}} B[\Omega],$$

assuming by inductive hypothesis that:

1. $\mathfrak{r}[\Omega,\Sigma] \Re_{\mathfrak{M}} A[\Omega] \to B[\Omega]$,
2. $\mathfrak{p}[\Omega,\Sigma] \Re_{\mathfrak{M}} A[\Omega]$.

This follows by Proposition 2 if:

$$(\lambda\gamma_1^{|A|_{\mathfrak{M}}\to|B|_{\mathfrak{M}}}.\lambda\gamma_2^{|A|_{\mathfrak{M}}}.\gamma_1\gamma_2)rp,$$

which β-reduces to

$$rp,$$

is a monadic realizer of $B[\Omega]$ for any inner realizers r and p of $A[\Omega] \to B[\Omega]$ and $A[\Omega]$ respectively. This follows immediately by definition of $\mathrm{R}_{\mathfrak{M}}$.

In the following cases we assume that Ω does not contain a substitution for the variable x and we write it explicitly when it is needed.

$\forall$I We have to show that:

$$\mathfrak{raise}_0(\lambda x^{\mathrm{Nat}}.\mathfrak{r}[\Omega,\Sigma]) \Re_{\mathfrak{M}} \forall x.\ A[\Omega],$$

assuming by inductive hypothesis that:

$$\mathfrak{r}[\Omega, x := \boldsymbol{n}, \Sigma] \Re_{\mathfrak{M}} A[\Omega, x := \boldsymbol{n}],$$

for any natural number n. This follows by Proposition 2 if:

$$(\lambda x^{\mathrm{Nat}}.\mathfrak{r}[\Omega,\Sigma]) \mathrm{R}_{\mathfrak{M}} \forall x.\ A[\Omega],$$

which by definition of $\mathrm{R}_{\mathfrak{M}}$ means that:

$$(\lambda x^{\mathrm{Nat}}.\mathfrak{r}[\Omega,\Sigma])\boldsymbol{n}\ \mathrm{R}_{\mathfrak{M}}\ A[\Omega, x := \boldsymbol{n}],$$

for any natural number n. By β-reducing we get:

$$\mathfrak{r}[\Omega, x := \boldsymbol{n}, \Sigma]\ \mathrm{R}_{\mathfrak{M}}\ A[\Omega, x := \boldsymbol{n}],$$

which holds by inductive hypothesis.

$\forall$E We have to show that:

$$(\mathfrak{star}_1(\lambda\gamma^{\mathrm{Nat}\to\|A\|_{\mathfrak{M}}}.\gamma(t[\Omega])))\mathfrak{r}[\Omega,\Sigma]\ \mathfrak{R}_{\mathfrak{M}}\ (A[x := t])[\Omega],$$

assuming by inductive hypothesis that:

$$\mathfrak{r}[\Omega,\Sigma]\ \mathfrak{R}_{\mathfrak{M}}\ \forall x.\ A[\Omega].$$

This follows by Proposition 1 if:

$$(\lambda\gamma^{\mathrm{Nat}\to\|A\|_{\mathfrak{M}}}.\gamma(t[\Omega])))r \rightsquigarrow r(t[\Omega]),$$

is a monadic realizer of $A[\Omega]$, for any inner realizer r of $\forall x.\ A[\Omega]$. This follows by definition of $\mathrm{R}_{\mathfrak{M}}$ for $r\ \mathrm{R}_{\mathfrak{M}}\ \forall x.\ A[\Omega]$, since $t[\Omega]$ is closed and thus reduces to a numeral.

$\exists$I We have to show that:

$$\mathfrak{raise}_1(\lambda\gamma^{|A|_{\mathfrak{M}}}.\,\mathsf{pair}\ t[\Omega]\gamma)\mathfrak{r}[\Omega,\Sigma]\ \mathfrak{R}_{\mathfrak{M}}\ \exists x.\ A[\Omega],$$

assuming by inductive hypothesis that:

$$\mathfrak{r}[\Omega,\Sigma]\ \mathfrak{R}_{\mathfrak{M}}\ A[\Omega, x := t].$$

This follows by Proposition 2 if:

$$(\lambda\gamma^{|A|_{\mathfrak{M}}}.\,\mathsf{pair}\ t[\Omega]\gamma)r \rightsquigarrow \mathsf{pair}\ t[\Omega]r$$

is an inner realizer of $\exists x.\ A[\Omega]$, for any inner realizer r of $A[\Omega, x := t]$. This follows by definition of $\mathrm{R}_{\mathfrak{M}}$.

$\exists$E We have to show that:

$$\mathfrak{star}_1(\lambda\gamma^{\mathrm{Nat}\times|A|_{\mathfrak{M}}}.(\lambda y^{\mathrm{Nat}}.\lambda\alpha^{|A|_{\mathfrak{M}}}.\mathfrak{r}_2[\Omega,\Sigma])(\mathsf{pr}_\mathrm{L}\,\gamma)(\mathsf{pr}_\mathrm{R}\,\gamma))\mathfrak{r}_1[\Omega,\Sigma]\ \mathfrak{R}_{\mathfrak{M}}\ C[\Omega],$$

assuming by inductive hypothesis that:

1. $\mathfrak{r}_1[\Omega,\Sigma]\ \mathfrak{R}_{\mathfrak{M}}\ \exists x.\, A[\Omega]$,
2. $\mathfrak{r}_2[\Omega, y := \boldsymbol{n}, \Sigma, \alpha := r]\ \mathfrak{R}_{\mathfrak{M}}\ C[\Omega]$, for any natural number n and any inner realizer r of $A[\Omega]$.

This follows by Proposition 1 and by the inductive hypothesis on $\mathfrak{r}_1$ if, for any inner realizer r_1 of $\exists x.\, A[\Omega]$:

$$\begin{aligned}
&(\lambda\gamma^{\mathrm{Nat}\times|A|_{\mathfrak{M}}}.(\lambda y^{\mathrm{Nat}}.\lambda\alpha^{|A|_{\mathfrak{M}}}.\mathfrak{r}_2[\Omega,\Sigma])(\mathsf{pr}_\mathrm{L}\,\gamma)(\mathsf{pr}_\mathrm{R}\,\gamma))r_1 \rightsquigarrow\\
&\rightsquigarrow (\lambda y^{\mathrm{Nat}}.\lambda\alpha^{|A|_{\mathfrak{M}}}.\mathfrak{r}_2[\Omega,\Sigma])(\mathsf{pr}_\mathrm{L}\,r_1)(\mathsf{pr}_\mathrm{R}\,r_1) \rightsquigarrow\\
&\rightsquigarrow ((\mathfrak{r}_2[\Omega,\Sigma])[y := \mathsf{pr}_\mathrm{L}\,r_1])[\alpha := \mathsf{pr}_\mathrm{R}\,r_1] \equiv\\
&\equiv \mathfrak{r}_2[\Omega, y := \mathsf{pr}_\mathrm{L}\,r_1, \Sigma, \alpha := \mathsf{pr}_\mathrm{R}\,r_1].
\end{aligned}$$

is a monadic realizer of $C[\Omega]$. By definition of $\mathrm{R}_{\mathfrak{M}}$ we have that $\mathsf{pr}_\mathrm{R}\,r_1\ \mathrm{R}_{\mathfrak{M}}\ A[x := \mathsf{pr}_\mathrm{L}\,r_1]$ and thus we can conclude by the inductive hypothesis on $\mathfrak{r}_2$.

Ind We have to show that:

$$(\mathfrak{raise}_0(\mathsf{crec}_\infty\,f))[\Omega,\Sigma]\ \mathfrak{R}_{\mathfrak{M}}\ (\forall x.\, A)[\Omega],$$

assuming that, for all naturals numbers n and for all $p : \mathrm{Nat} \to T(\mathrm{Unit} \to T|A|_{\mathfrak{M}})$ such that $p\ \mathrm{R}_{\mathfrak{M}}\ \forall z.\, z < \boldsymbol{n} \to A[x := z]$:

$$r[\Omega, y := \boldsymbol{n}, \Sigma, \alpha_{k+1}] := p]\ \mathfrak{R}_{\mathfrak{M}}\ A[x := y][\Omega, y := \boldsymbol{n}].$$

Note that $A[x := y][\Omega, y := \boldsymbol{n}]$ is just $A[\Omega, x := \boldsymbol{n}]$. By Proposition 2 we get the conclusion if $\mathsf{crec}_\infty\,f[\Omega,\Sigma]\ \mathrm{R}_{\mathfrak{M}}\ \forall x.\, A[\Omega]$, which by definition of $\mathrm{R}_{\mathfrak{M}}$ means that

$$\mathsf{crec}_\infty\,f[\Omega,\Sigma]\boldsymbol{n}\ \mathfrak{R}_{\mathfrak{M}}\ A[\Omega, x := \boldsymbol{n}]$$

for any natural number n. In order to show this we shall prove that for any natural number n and any $\omega \in \mathbb{N} \cup \{\infty\}$ such that either $\omega = \infty$ or $\omega > n$, we have:

$$\mathsf{crec}_\omega\,f[\Omega,\Sigma]\boldsymbol{n}\ \mathfrak{R}_{\mathfrak{M}}\ A[\Omega, x := \boldsymbol{n}].$$

We proceed by complete induction on n, so we assume that the statement holds for all natural numbers m such that $m < n$. We begin by reducing the realizer (in the first step we use the assumption on ω:

$$\begin{aligned}\mathsf{crec}_\omega\, f[\Omega,\Sigma]\boldsymbol{n} &\rightsquigarrow f[\Omega,\Sigma]\boldsymbol{n}(\mathsf{crec}_n\, f[\Omega,\Sigma])\\ &\rightsquigarrow (\lambda\alpha.\mathfrak{r}[\Omega, y := \boldsymbol{n}])(\lambda z^{\mathrm{Nat}}.\,\mathfrak{raise}_0(\lambda\_^{\mathrm{Unit}}.\,\mathsf{crec}_n\, f[\Omega,\Sigma]z))\\ &\rightsquigarrow \mathfrak{r}[\Omega, y := \boldsymbol{n}, \Sigma, \alpha := \lambda z^{\mathrm{Nat}}.\,\mathfrak{raise}_0(\lambda\_^{\mathrm{Unit}}.\,\mathsf{crec}_n\, f[\Omega,\Sigma]z)]\end{aligned}$$

Then we have to show that:

$$\mathfrak{r}[\Omega, y := \boldsymbol{n}, \Sigma, \alpha := \lambda z^{\mathrm{Nat}}.\,\mathfrak{raise}_0(\lambda\_^{\mathrm{Unit}}.\,\mathsf{crec}_n\, f[\Omega,\Sigma]z)]\ \mathfrak{R}_{\mathfrak{M}}\ A[\Omega, x := \boldsymbol{n}].$$

This follows from the inductive hypothesis on the premise of the complete induction rule if we can show that:

$$\lambda z^{\mathrm{Nat}}.\,\mathfrak{raise}_0(\lambda\_^{\mathrm{Unit}}.\,\mathsf{crec}_n\, f[\Omega,\Sigma]z)\ \mathrm{R}_{\mathfrak{M}}\ \forall z.\ z < \boldsymbol{n} \to A[x := z].$$

By definition of $\mathrm{R}_{\mathfrak{M}}$ this is the case if:

$$\mathfrak{raise}_0(\lambda\_^{\mathrm{Unit}}.\,\mathsf{crec}_n\, f[\Omega,\Sigma]\boldsymbol{m})\ \mathfrak{R}_{\mathfrak{M}}\ \boldsymbol{m} < \boldsymbol{n} \to A[x := \boldsymbol{m}],$$

for all natural numbers m. By Property MR1 this follows from:

$$\lambda\_^{\mathrm{Unit}}.\,\mathsf{crec}_n\, f[\Omega,\Sigma]\boldsymbol{m}\ \mathrm{R}_{\mathfrak{M}}\ \boldsymbol{m} < \boldsymbol{n} \to A[x := \boldsymbol{m}].$$

Again by definition of $\mathrm{R}_{\mathfrak{M}}$ this is equivalent to showing that for any u : Unit such that $u\ \mathrm{R}_{\mathfrak{M}}\ \boldsymbol{m} < \boldsymbol{n}$ we have:

$$\mathsf{crec}_n\, f[\Omega,\Sigma]\boldsymbol{m}\ \mathrm{R}_{\mathfrak{M}}\ A[x := \boldsymbol{m}].$$

Note that, since u : Unit, $u \rightsquigarrow *$, so there are two possible cases: either $m < n$ is true and then $u\ \mathrm{R}_{\mathfrak{M}}\ \boldsymbol{m} <\mathrm{R}_{\mathfrak{M}}\ \boldsymbol{n}$ for any u : Unit or $m < n$ is false and no u : Unit can realize $\boldsymbol{m} < \boldsymbol{n}$. In both cases the statement holds: in the former case by inductive hypothesis on m and in the latter case trivially since the universal quantification on u is empty.

□

Theorem 1 entails that any specific monadic realizability notion is a sound semantics for at least HA. Later, when we prove that HA + EM_1 is sound with respect to interactive realizability semantics, we will only need to show that EM_1 is sound since the soundness of HA derives from Theorem 1.

3.3 Monadic Interactive Realizability

In this section we define interactive realizability as a particular notion of monadic realizability. Thus we show that monadic realizability may realize a sub-classical principle, in this case excluded middle restricted to semi-decidable statements.

3.3.1 A Syntactic Monad for Interactive Realizability

In order to describe the computational properties of interactive realizability (see [2]) we need to define a suitable monad. As we said, interactive realizability is based on the idea of learning by trial and error. We express the idea of trial and error with an exception monad: a term of intended type X has actual type $X + \mathrm{Ex}$, where Ex is the type of exceptions, so that a computation may either return its intended value or an exception. The learning part, which is described by the dependency on a knowledge state, fits with a part of the side-effects monad (see [18] for more details): a term of intended type X has actual type State $\to X$, where State is the type of knowledge states, so that the value of a computation may change with the state. The syntactic monad we are about to define for interactive realizability combines these two monads.

We need to extend system T' with two base types State and Ex and a term constant that "merges" two exceptions into one:

$$\mathsf{merge} : \mathrm{Ex} \to \mathrm{Ex} \to \mathrm{Ex}.$$

We shall avoid defining a specific syntax for terms of type State and Ex. Instead we exhibit their intended interpretation and, using this interpretation as a guide, we shall require some properties on reductions involving them.

We write $\mathcal{R}_k$ for the set of symbols of the k-ary predicates in HA. The intended interpretation of a (knowledge) state s is a partial function

$$[\![s]\!] : \left(\bigcup_{k=0}^{\infty} \mathcal{R}_{k+1} \times \mathbb{N}^k \right) \rightharpoonup \mathbb{N},$$

that sends a $k+1$-ary predicate symbol P and a k-tuple of parameters $m_1, \ldots, m_k \in \mathbb{N}$ to a witness for $\exists x.\ P(\boldsymbol{m}_1, \ldots, \boldsymbol{m}_k, x)$. We interpret the fact that a state s is undefined for some $P, m_1, \ldots, m_k$ as a lack of knowledge about a suitable witness. This is either due to the state

being incomplete, meaning that there exists a suitable witness m we could use to extend the state by setting $s(P, (m_1, \ldots, m_k)) = m$, or to the fact that there are no suitable witness, meaning that $\forall x.\ \neg P(\boldsymbol{m}_1, \ldots, \boldsymbol{m}_k, x)$ holds[3]. We require that s satisfies two properties. The first is for s to be *sound*, meaning that its values are actually witnesses. More precisely:

$$[\![s]\!](P, (m_1, \ldots, m_k)) = m \text{ entails } P(\boldsymbol{m}_1, \ldots, \boldsymbol{m}_k, \boldsymbol{m}).$$

The second is that s is *finite*, namely that the domain of s (the set of values s is defined on) is finite. This because we want a knowledge state to encode a finite quantity of information. Let $[\![\text{State}]\!]$, the set of all finite sound states, be the intended interpretation of the type State. Recall that there is a canonical partial order on states given by the extension relation: we write $s_1 \leq s_2$ and read "s_2 *extends* s_1" if and only if s_2 is defined whenever s_1 is and with the same value.

An exception e : Ex is produced when we instantiate an assumption of the form $\forall x.\ \neg P(\boldsymbol{m}_1, \ldots, \boldsymbol{m}_k, x)$ with some m such that $\neg P(\boldsymbol{m}_1, \ldots, \boldsymbol{m}_k, \boldsymbol{m})$ does not actually hold (remember that we proceed by trial and error, in particular we may assume things that are actually false). This means that m is a witness for $\exists x.\ P(\boldsymbol{m}_1, \ldots, \boldsymbol{m}_k, x)$, in particular it could be used to extend the knowledge state on values where it was previously undefined. The role of exceptions is to encode information about the discovery of new witnesses: since we use this information to extend states the intended interpretation of an exception e is as a partial function:

$$[\![e]\!] : [\![\text{State}]\!] \rightharpoonup [\![\text{State}]\!].$$

Since e extends states we require that $s \leq e(s)$. We interpret an exception as a partial function because an exception e may fail to extend some state s. The reason is that e may contain information about a witness m' for an existential statement $\exists x.\ P(\boldsymbol{m}_1, \ldots, \boldsymbol{m}_k, x)$ on which s is already defined as m. Note that an existential formula can have more that one witness so two cases may arise: either $m = m'$, meaning that the information of e is already part of s or $m \neq m'$ so that the information of e is incompatible with the information of the state. In the first case $e(s) = s$, while in the second case $e(s)$ is not defined.

[3]Here we are using EM_1 at the metalevel in order to explain the possible situations. Using a principle at the metalevel in order to justify the same principle in the logic is a common practice. In our treatment this is not problematic because we never claim to be able to effectively decide which situation we are in.

Before defining the syntactic monad $\Im R$ for interactive realizability, we need to introduce some terminology on exceptions and states.

Definition 8 (Terminology on Exceptions and States)**.** *We say that a term of type* $X + \mathrm{Ex}$ *is either a* regular *value* a *if it reduces to* $\mathsf{in}_L\, a$ *for some term* $a : X$ *or an* exceptional *value if it reduces to* $\mathsf{in}_R\, e$ *for some term* $e : \mathrm{Ex}$*. We say that a term of type* State $\to X$ *is a* state function*. Finally we say that an exception* e properly extends s *if* $e(s)$ *is defined and* $s < e(s)$*.*

Note that different exceptions might be used to extend a knowledge state in incompatible ways, that is, by sending the same predicate symbol and the same tuple of parameters into different witnesses. The role of the merge function is to put together the information from two exceptions into a single exception. This means that merge cannot simply put together all the information from its argument: if such information contains more that one distinct witness for the same existential statement it must choose one in some arbitrary way, for instance the leftmost or the minimum witness. Many choices for merge are possible, provided that they satisfy the following property:

$$\left.\begin{array}{l} e_1 \text{ properly extends } s \\ e_2 \text{ properly extends } s \end{array}\right\} \text{entails that } \mathsf{merge}\, e_1 e_2 \text{ properly extends } s, \tag{EX}$$

for any state s and exceptions e_1, e_2. Simple choices for merge are the projections, always selecting the first or the second argument, or any combination of them using an arbitrary criterion to select which value to return. Of course, in general there is no need for $\mathsf{merge}\, e_1 e_2$ to be e_1 or e_2.

Before the definition we give an informal description of $\Im R$. The monad $\Im R$ maps a type X to State $\to (X + \mathrm{Ex})$, that is, values of type X are lifted to state functions that can throw exceptions. The term $\mathfrak{unit}_{\Im R}$ maps a value $a : X$ to a constant state function that returns the regular value a. If $f : X \to T_{\Im R} Y$ then $\mathfrak{star}_{\Im R}\, f$ is a function with two arguments, a state s and a state function $\mathfrak{a} : T_{\Im R} X$. It evaluates $\mathfrak{a}$ on s: if this results in a regular value $a : X$ it applies f to a, otherwise it propagates the exceptional value. Lastly, if $\mathfrak{a} : T_{\Im R} X$ and $\mathfrak{b} : T_{\Im R} Y$ are two state functions, then $\mathfrak{merge}_{\Im R}\, \mathfrak{a}\mathfrak{b}$ is a state function that evaluates its arguments on its state argument: when both arguments are regular values it returns their pair, otherwise it propagates the exception(s), using merge if both arguments are exceptional values.

We are now ready to give the formal definition of $\Im R$.

Definition 9 (Interactive Realizability Monad)**.** *Let* $\Im R$ *be the tuple* $(T_{\Im R}, \mathfrak{unit}_{\Im R}, \mathfrak{star}_{\Im R}, \mathfrak{merge}_{\Im R})$*, where*

$$
\begin{aligned}
T_{\Im R}X &= \mathrm{State} \to (X + \mathrm{Ex}),\\
\mathfrak{unit}^{X}_{\Im R} &\equiv \lambda x^{X}.\lambda\_^{\mathrm{State}}.\, \mathit{in}^{X,\mathrm{Ex}}_{L}\, x,\\
\mathfrak{star}^{X,Y}_{\Im R} &\equiv \lambda f^{X\to T_{\Im R}Y}.\lambda \mathfrak{x}^{T_{\Im R}X}.\lambda s^{\mathrm{State}}.\, \mathit{case}^{X,\mathrm{Ex},Y+\mathrm{Ex}}(\mathfrak{x}s)(\lambda x^{X}.f x s)\, \mathit{in}^{Y,\mathrm{Ex}}_{R},\\
\mathfrak{merge}^{X,Y}_{\Im R} &\equiv \lambda \mathfrak{x}^{T_{\Im R}X}.\lambda \mathfrak{y}^{T_{\Im R}Y}.\lambda s^{\mathrm{State}}.\, \mathit{case}^{X,\mathrm{Ex},(X\times Y)+\mathrm{Ex}}(\mathfrak{x}s)\\
&\qquad (\lambda x^{X}.\, \mathit{case}^{Y,\mathrm{Ex},(X\times Y)+\mathrm{Ex}}(\mathfrak{y}s)(\lambda y^{Y}.\, \mathit{in}^{X\times Y,\mathrm{Ex}}_{L}(\mathit{pair}\, x y))\, \mathit{in}^{X\times Y,\mathrm{Ex}}_{R})\\
&\qquad (\lambda e_1^{\mathrm{Ex}}.\, \mathit{case}^{Y,\mathrm{Ex},(X\times Y)+\mathrm{Ex}}(\mathfrak{y}s)(\lambda\_^{Y}.\, \mathit{in}^{X\times Y,\mathrm{Ex}}_{R}\, e_1)(\lambda e_2^{\mathrm{Ex}}.\, \mathit{in}^{X\times Y,\mathrm{Ex}}_{R}(\mathit{merge}\, e_1 e_2))),
\end{aligned}
$$

for some merge *satisfying Property EX.*

The term $\mathfrak{unit}^{X}_{\Im R}$ takes a value $a : X$ and produces a constant state function that returns the regular (as opposed to exceptional) value a. The term $\mathfrak{star}^{X,Y}_{\Im R}$ takes a function $f : X \to T_{\Im R}Y$ and returns a function f' which lifts the domain of f to $T_{\Im R}X$. The state function returned by f' when applied to some $\mathfrak{a} : T_{\Im R}X$ behaves as follows: it evaluates $\mathfrak{a}$ on the state and if $\mathfrak{a}s$ is a regular value $a : X$ it returns fa; otherwise if $\mathfrak{a}s$ is an exception it simply propagates the exception.

$\mathfrak{a}s$	$\mathfrak{b}s$
a	fa
e	e

The term $\mathfrak{merge}^{X,Y}_{\Im R}$ takes two state functions $\mathfrak{a} : T_{\Im R}X$ and $\mathfrak{b} : T_{\Im R}Y$ and returns a state function $\mathfrak{c} : T_{\Im R}(X \times Y)$. When both arguments are regular values it returns their pair, otherwise it propagates the exception(s), using merge if both arguments are exceptional.

$\mathfrak{a}s$	$\mathfrak{b}s$	$\mathfrak{c}s$
a	b	pair ab
e_1	b	e_1
a	e_2	e_2
e_1	e_2	merge $e_1 e_2$

We still need to check that Definition 9 is correct and that $\Im R$ really is a syntactic monad.

Proposition 3 (The Syntactic Monad $\Im R$)**.** *$\Im R$ is a syntactic monad.*

Proof. We just need to check that $\mathfrak{unit}_{\Im R}$, $\mathfrak{star}_{\Im R}$ and $\mathfrak{merge}_{\Im R}$ satisfy all the properties in Definition 4. This amounts to perform some reductions.

M1 Given any $\mathfrak{x} : T_{\Im R}X$, we have:

$$
\begin{aligned}
&\mathfrak{star}^{X,X}_{\Im R}\,\mathfrak{unit}^{X}_{\Im R}\,\mathfrak{x}\\
&\equiv (\lambda f^{X\to T_{\Im R}X}.\lambda \mathfrak{x}^{T_{\Im R}X}.\lambda s^{\mathrm{State}}.\,\mathsf{case}^{X,\mathrm{Ex},X+\mathrm{Ex}}(\mathfrak{x}s)(\lambda x^{X}.fxs)\,\mathsf{in}^{X,\mathrm{Ex}}_{\mathrm{R}})\,\mathfrak{unit}^{X}_{\Im R}\,\mathfrak{x}\\
&\to_\beta (\lambda \mathfrak{x}^{T_{\Im R}X}.\lambda s^{\mathrm{State}}.\,\mathsf{case}^{X,\mathrm{Ex},X+\mathrm{Ex}}(\mathfrak{x}s)(\lambda x^{X}.\,\mathfrak{unit}^{X}_{\Im R}\,xs)\,\mathsf{in}^{X,\mathrm{Ex}}_{\mathrm{R}})\mathfrak{x}\\
&\to_\beta \lambda s^{\mathrm{State}}.\,\mathsf{case}^{X,\mathrm{Ex},X+\mathrm{Ex}}(\mathfrak{x}s)(\lambda x^{X}.\,\mathfrak{unit}^{X}_{\Im R}\,xs)\,\mathsf{in}^{X,\mathrm{Ex}}_{\mathrm{R}}\\
&\equiv \lambda s^{\mathrm{State}}.\,\mathsf{case}^{X,\mathrm{Ex},X+\mathrm{Ex}}(\mathfrak{x}s)(\lambda x^{X}.(\lambda x^{X}.\lambda \_^{\mathrm{State}}.\,\mathsf{in}^{X,\mathrm{Ex}}_{\mathrm{L}}\,x)xs)\,\mathsf{in}^{X,\mathrm{Ex}}_{\mathrm{R}}\\
&\to_\beta \lambda s^{\mathrm{State}}.\,\mathsf{case}^{X,\mathrm{Ex},X+\mathrm{Ex}}(\mathfrak{x}s)(\lambda x^{X}.(\lambda \_^{\mathrm{State}}.\,\mathsf{in}^{X,\mathrm{Ex}}_{\mathrm{L}}\,x)s)\,\mathsf{in}^{X,\mathrm{Ex}}_{\mathrm{R}}\\
&\to_\beta \lambda s^{\mathrm{State}}.\,\mathsf{case}^{X,\mathrm{Ex},X+\mathrm{Ex}}(\mathfrak{x}s)(\lambda x^{X}.\,\mathsf{in}^{X,\mathrm{Ex}}_{\mathrm{L}}\,x)\,\mathsf{in}^{X,\mathrm{Ex}}_{\mathrm{R}}\\
&=_\eta \lambda s^{\mathrm{State}}.\,\mathsf{case}^{X,\mathrm{Ex},X+\mathrm{Ex}}(\mathfrak{x}s)\,\mathsf{in}^{X,\mathrm{Ex}}_{\mathrm{L}}\,\mathsf{in}^{X,\mathrm{Ex}}_{\mathrm{R}}\\
&=_\times \lambda s^{\mathrm{State}}.\mathfrak{x}s\\
&=_\eta \mathfrak{x},
\end{aligned}
$$

as required by Property M1.

M2 Given any $f : X \to T_{\Im R}Y$ and $x : X$, we have:

$$
\begin{aligned}
&\mathfrak{star}^{X,Y}_{\Im R}\,f(\mathfrak{unit}^{X}_{\Im R}\,x)\\
&\equiv (\lambda f^{X\to T_{\Im R}Y}.\lambda \mathfrak{x}^{T_{\Im R}X}.\lambda s^{\mathrm{State}}.\,\mathsf{case}^{X,\mathrm{Ex},Y+\mathrm{Ex}}(\mathfrak{x}s)(\lambda x^{X}.fxs)\,\mathsf{in}^{Y,\mathrm{Ex}}_{\mathrm{R}})f(\mathfrak{unit}^{X}_{\Im R}\,x)\\
&\to_\beta (\lambda \mathfrak{x}^{T_{\Im R}X}.\lambda s^{\mathrm{State}}.\,\mathsf{case}^{X,\mathrm{Ex},Y+\mathrm{Ex}}(\mathfrak{x}s)(\lambda x^{X}.fxs)\,\mathsf{in}^{Y,\mathrm{Ex}}_{\mathrm{R}})(\mathfrak{unit}^{X}_{\Im R}\,x)\\
&\to_\beta \lambda s^{\mathrm{State}}.\,\mathsf{case}^{X,\mathrm{Ex},Y+\mathrm{Ex}}(\mathfrak{unit}^{X}_{\Im R}\,xs)(\lambda x^{X}.fxs)\,\mathsf{in}^{Y,\mathrm{Ex}}_{\mathrm{R}}\\
&\equiv \lambda s^{\mathrm{State}}.\,\mathsf{case}^{X,\mathrm{Ex},Y+\mathrm{Ex}}((\lambda x^{X}.\lambda \_^{\mathrm{State}}.\,\mathsf{in}^{X,\mathrm{Ex}}_{\mathrm{L}}\,x)xs)(\lambda x^{X}.fxs)\,\mathsf{in}^{Y,\mathrm{Ex}}_{\mathrm{R}}\\
&\to_\beta \lambda s^{\mathrm{State}}.\,\mathsf{case}^{X,\mathrm{Ex},Y+\mathrm{Ex}}(\mathsf{in}^{X,\mathrm{Ex}}_{\mathrm{L}}\,x)(\lambda x^{X}.fxs)\,\mathsf{in}^{Y,\mathrm{Ex}}_{\mathrm{R}}\\
&\to_\times \lambda s^{\mathrm{State}}.(\lambda x^{X}.fxs)x\\
&\to_\beta \lambda s^{\mathrm{State}}.fxs\\
&=_\eta fx,
\end{aligned}
$$

as required by Property M2.

M3 Given any $x : X$ and $y : Y$, we have:

$$
\begin{aligned}
&\mathfrak{merge}_{\mathfrak{I}R}(\mathfrak{unit}_{\mathfrak{I}R}\, x)(\mathfrak{unit}_{\mathfrak{I}R}\, y)\\
&\quad\equiv (\lambda \mathfrak{x}^{T_{\mathfrak{I}R}X}.\lambda \mathfrak{y}^{T_{\mathfrak{I}R}Y}.\lambda s^{\mathrm{State}}.\, \mathsf{case}^{X,\mathrm{Ex},(X\times Y)+\mathrm{Ex}}(\mathfrak{x}s)\\
&\qquad (\lambda x^{X}.\, \mathsf{case}^{Y,\mathrm{Ex},(X\times Y)+\mathrm{Ex}}(\mathfrak{y}s)(\lambda y^{Y}.\, \mathsf{in}_{\mathrm{L}}^{X\times Y,\mathrm{Ex}}(\mathsf{pair}\, xy))\, \mathsf{in}_{\mathrm{R}}^{X\times Y,\mathrm{Ex}})\\
&\qquad (\lambda e_1^{\mathrm{Ex}}.\, \mathsf{case}^{Y,\mathrm{Ex},(X\times Y)+\mathrm{Ex}}(\mathfrak{y}s)(\lambda x^{Y}.\, \mathsf{in}_{\mathrm{R}}^{X\times Y,\mathrm{Ex}}\, e_1)(\lambda e_2^{\mathrm{Ex}}.\, \mathsf{in}_{\mathrm{R}}^{X\times Y,\mathrm{Ex}}(\mathsf{merge}\, e_1 e_2))))\\
&\qquad (\mathfrak{unit}_{\mathfrak{I}R}\, x)(\mathfrak{unit}_{\mathfrak{I}R}\, y)\\
&\quad\to_{\beta} \lambda s^{\mathrm{State}}.\, \mathsf{case}^{X,\mathrm{Ex},(X\times Y)+\mathrm{Ex}}(\mathfrak{unit}_{\mathfrak{I}R}\, xs)\\
&\qquad (\lambda x^{X}.\, \mathsf{case}^{Y,\mathrm{Ex},(X\times Y)+\mathrm{Ex}}(\mathfrak{unit}_{\mathfrak{I}R}\, ys)(\lambda y^{Y}.\, \mathsf{in}_{\mathrm{L}}^{X\times Y,\mathrm{Ex}}(\mathsf{pair}\, xy))\, \mathsf{in}_{\mathrm{R}}^{X\times Y,\mathrm{Ex}})(\ldots)\\
&\quad\equiv \lambda s^{\mathrm{State}}.\, \mathsf{case}^{X,\mathrm{Ex},(X\times Y)+\mathrm{Ex}}((\lambda x^{X}.\lambda \_^{\mathrm{State}}.\, \mathsf{in}_{\mathrm{L}}^{X,\mathrm{Ex}}\, x)xs)\\
&\qquad (\lambda x^{X}.\, \mathsf{case}^{Y,\mathrm{Ex},(X\times Y)+\mathrm{Ex}}(\mathfrak{unit}_{\mathfrak{I}R}\, ys)(\lambda y^{Y}.\, \mathsf{in}_{\mathrm{L}}^{X\times Y,\mathrm{Ex}}(\mathsf{pair}\, xy))\, \mathsf{in}_{\mathrm{R}}^{X\times Y,\mathrm{Ex}})(\ldots)\\
&\quad\to_{\beta} \lambda s^{\mathrm{State}}.\, \mathsf{case}^{X,\mathrm{Ex},(X\times Y)+\mathrm{Ex}}(\mathsf{in}_{\mathrm{L}}^{X,\mathrm{Ex}}\, x)\\
&\qquad (\lambda x^{X}.\, \mathsf{case}^{Y,\mathrm{Ex},(X\times Y)+\mathrm{Ex}}(\mathfrak{unit}_{\mathfrak{I}R}\, ys)(\lambda y^{Y}.\, \mathsf{in}_{\mathrm{L}}^{X\times Y,\mathrm{Ex}}(\mathsf{pair}\, xy))\, \mathsf{in}_{\mathrm{R}}^{X\times Y,\mathrm{Ex}})(\ldots)\\
&\quad\to_{+} \lambda s^{\mathrm{State}}.\, \mathsf{case}^{Y,\mathrm{Ex},(X\times Y)+\mathrm{Ex}}(\mathfrak{unit}_{\mathfrak{I}R}\, ys)(\lambda y^{Y}.\, \mathsf{in}_{\mathrm{L}}^{X\times Y,\mathrm{Ex}}(\mathsf{pair}\, xy))\, \mathsf{in}_{\mathrm{R}}^{X\times Y,\mathrm{Ex}}\\
&\quad\equiv \lambda s^{\mathrm{State}}.\, \mathsf{case}^{Y,\mathrm{Ex},(X\times Y)+\mathrm{Ex}}((\lambda y^{Y}.\lambda \_^{\mathrm{State}}.\, \mathsf{in}_{\mathrm{L}}^{Y,\mathrm{Ex}}\, y)ys)(\lambda y^{Y}.\, \mathsf{in}_{\mathrm{L}}^{X\times Y,\mathrm{Ex}}(\mathsf{pair}\, xy))\, \mathsf{in}_{\mathrm{R}}^{X\times Y,\mathrm{Ex}}\\
&\quad\to_{\beta} \lambda s^{\mathrm{State}}.\, \mathsf{case}^{Y,\mathrm{Ex},(X\times Y)+\mathrm{Ex}}(\mathsf{in}_{\mathrm{L}}^{Y,\mathrm{Ex}}\, y)(\lambda y^{Y}.\, \mathsf{in}_{\mathrm{L}}^{X\times Y,\mathrm{Ex}}(\mathsf{pair}\, xy))\, \mathsf{in}_{\mathrm{R}}^{X\times Y,\mathrm{Ex}}\\
&\quad\to_{+} \lambda s^{\mathrm{State}}.(\lambda y^{Y}.\, \mathsf{in}_{\mathrm{L}}^{X\times Y,\mathrm{Ex}}(\mathsf{pair}\, xy))y\\
&\quad\to_{\beta} \lambda s^{\mathrm{State}}.\, \mathsf{in}_{\mathrm{L}}^{X\times Y,\mathrm{Ex}}(\mathsf{pair}\, xy)\\
&\quad\equiv \mathfrak{unit}_{\mathfrak{I}R}^{X\times Y}(\mathsf{pair}\, xy),
\end{aligned}
$$

as required by Property M3.

□

3.3.2 The Interactive Realizability Semantics

We now define a family of monadic realizability relations, one for each state s, requiring that a realizer, applied to a knowledge state s, either realizes a formula in the sense of the BHK semantics or can extend s with new knowledge.

Definition 10 (Interactive Realizability Relation)**.** *Let s be a state, $\mathfrak{r} : \|A\|_{\mathfrak{I}R}$ be a term and A a closed formula. We define two realizability relations $\mathfrak{R}^s_{\mathfrak{I}R}$ and $\mathrm{R}^s_{\mathfrak{I}R}$ by simultaneous*

induction on the structure of A:

- $\mathfrak{r} \ \Re^{s}_{\Im R} \ A$ *if and only if we have that* $\mathfrak{r}s$ *is either a regular value* r *such that* $r \ \mathrm{R}^{s}_{\Im R} \ A$ *or an exceptional value* e *such that* e *properly extends* s*,*

- $\mathrm{R}^{s}_{\Im R}$ *is defined in terms of* $\Re^{s}_{\Im R}$ *by the clauses in Definition 6.*

We say that $\mathfrak{r}$ *(resp.* r*) is a* monadic *(resp.* inner*)* interactive realizer *of* A *with respect to* s *when* $\mathfrak{r} : ||A||_{\Im R}$ *(resp.* $r : |A|_{\Im R}$*) and* $\mathfrak{r} \ \Re^{s}_{\Im R} \ A$ *(resp.* $r \ \mathrm{R}^{s}_{\Im R} \ A$*).*

In order to show that any interactive realizability relations with respect to a state is a monadic realizability relation we need to verify that is satisfies the required properties.

Proposition 4 (The Monadic Realizability Relation $\Re^{s}_{\Im R}$)**.** *For any state* s*,* $\Re^{s}_{\Im R}$ *is a monadic realizability relation.*

Proof. Let s be any state. We have to show that $\Re^{s}_{\Im R}$ satisfies the properties in Definition 6.

MR1 We begin with Property MR1, namely, for any inner interactive realizer r of a formula A with respect to s, we show that:

$$\mathfrak{unit}_{\Im R}\, r \ \Re^{s}_{\Im R} \ A.$$

By unfolding the definition of $\mathfrak{unit}_{\Im R}$ we have that:

$$\begin{aligned} \mathfrak{unit}_{\Im R}\, r s &\rightsquigarrow (\lambda \_^{\mathrm{State}}.\, \mathsf{in}_{\mathrm{L}}\, r)s \\ &\rightsquigarrow \mathsf{in}_{\mathrm{L}}\, r, \end{aligned}$$

thus, by definition of $\Re^{s}_{\Im R}$, we have to check that $r \ \mathrm{R}^{s}_{\Im R} \ A$, which holds by assumption.

MR2 In order to show Property MR2, for any formulas A and B, we take an inner interactive realizer r of $A \to B$ with respect to s, that is, a term $r : |A|_{\Im R} \to ||B||_{\Im R}$ such that rp is a monadic interactive realizer of B with respect to s, for any inner interactive realizer p of A with respect to s. Then we have to show that, given a monadic interactive realizer $\mathfrak{p}$ of A with respect to s, we have:

$$\mathfrak{star}_{\Im R}\, r\mathfrak{p} \ \Re^{s}_{\Im R} \ B.$$

By definition of $\mathfrak{R}^s_{\mathfrak{I}R}$ we apply s to the realizer and by unfolding the definition of $\mathfrak{star}_{\mathfrak{I}R}$ and reducing we get:

$$\mathfrak{star}_{\mathfrak{I}R}\, r\mathfrak{p}s \rightsquigarrow \mathsf{case}(\mathfrak{p}s)(\lambda x^{|A|_{\mathfrak{I}R}}.rxs)\,\mathsf{in}_{\mathrm{R}}\,. \tag{3.2}$$

Since $\mathfrak{p}\ \mathfrak{R}^s_{\mathfrak{I}R}\ A$, we know that $\mathfrak{p}s$ reduces to either a regular value $\mathsf{in}_{\mathrm{L}}\,p$, for some inner realizer p of A with respect to s, or an exceptional value $\mathsf{in}_{\mathrm{R}}\,e$, for some exception e that properly extends s.

- In the former case, (3.2) reduces to rps. By the assumptions we made on r and p, rp is a monadic interactive realizer of B with respect to s, and thus rps reduces to either a regular value which is an inner interactive realizer of B with respect to s or an exceptional value which properly extends s. Thus $\mathfrak{star}_{\mathfrak{I}R}\, r\mathfrak{p}$ is a monadic interactive realizer of B with respect to s as required.
- In the latter case, (3.2) reduces to $\mathsf{in}_{\mathrm{R}}\,e$. Since e properly extends s, $\mathfrak{star}_{\mathfrak{I}R}\, r\mathfrak{p}$ is again a monadic interactive realizer of B with respect to s as required.

MR3 Finally we have to show Property MR3. We assume that $\mathfrak{p}$ and $\mathfrak{q}$ are monadic interactive realizers of A and B respectively, both with respect to s. Then we have to show that:

$$\mathfrak{merge}_{\mathfrak{I}R}\, \mathfrak{p}\mathfrak{q}\ \mathfrak{R}^s_{\mathfrak{I}R}\ A \wedge B.$$

By definition of $\mathfrak{R}^s_{\mathfrak{I}R}$, this means we have to show that

$$\mathfrak{merge}_{\mathfrak{I}R}\, \mathfrak{p}\mathfrak{q}s$$

reduces to either a regular value which is an inner interactive realizers Since $\mathfrak{p}$ and $\mathfrak{q}$ are monadic interactive realizers, $\mathfrak{p}s$ and $\mathfrak{q}s$ either reduce to regular values $\mathsf{in}_{\mathrm{L}}\,p$ and $\mathsf{in}_{\mathrm{L}}\,q$, where p and q are inner interactive realizers of respectively A and B with respect to s, or to exceptional values $\mathsf{in}_{\mathrm{R}}\,e_1$ and $\mathsf{in}_{\mathrm{R}}\,e_2$, where e_1 and e_2 properly extend s. By unfolding the definition of $\mathfrak{merge}_{\mathfrak{I}R}$ and reducing we get:

$$\begin{aligned}\mathfrak{merge}_{\mathfrak{I}R}\, \mathfrak{p}\mathfrak{q}s \rightsquigarrow\ & \mathsf{case}(\mathfrak{p}s)(\lambda x^{|A|_{\mathfrak{I}R}}.\,\mathsf{case}(\mathfrak{q}s)(\lambda y^{|B|_{\mathfrak{I}R}}.\,\mathsf{in}_{\mathrm{L}}(\mathsf{pair}\,xy))\,\mathsf{in}_{\mathrm{R}}) \\ & (\lambda e_1^{\mathrm{Ex}}.\,\mathsf{case}(\mathfrak{q}s)(\lambda\_^{|B|_{\mathfrak{I}R}}.\,\mathsf{in}_{\mathrm{R}}\,e_1)(\lambda e_2^{\mathrm{Ex}}.\,\mathsf{in}_{\mathrm{R}}(\mathsf{merge}\,e_1e_2)))\end{aligned} \tag{3.3}$$

We distinguish four cases depending on how $\mathfrak{p}s$ and $\mathfrak{q}s$ reduce:

$\mathfrak{p}s \rightsquigarrow$ **$\mathsf{in_L}$ p and $\mathfrak{q}s \rightsquigarrow$ $\mathsf{in_L}$ q** In this case (3.3) reduces as follows:

$$\begin{aligned}\mathfrak{merge}_{\Im R}\,\mathfrak{p}\mathfrak{q}s &\rightsquigarrow \mathsf{case}(\mathfrak{q}s)(\lambda y^{|B|_{\Im R}}.\,\mathsf{in_L}(\mathsf{pair}\,py))\,\mathsf{in_R}\\ &\rightsquigarrow \mathsf{in_L}(\mathsf{pair}\,pq).\end{aligned}$$

Since it is a regular value, we have to show that $\mathsf{pair}\,pq\ \mathsf{R}^s_{\Im R}\ A \wedge B$. This follows by definition of $\mathsf{R}^s_{\Im R}$ and from the assumption that $p\ \mathsf{R}^s_{\Im R}\ A$ and $q\ \mathsf{R}^s_{\Im R}\ B$.

$\mathfrak{p}s \rightsquigarrow$ **$\mathsf{in_L}$ p and $\mathfrak{q}s \rightsquigarrow$ $\mathsf{in_R}$ e_2** In this case (3.3) reduces as follows:

$$\begin{aligned}\mathfrak{merge}_{\Im R}\,\mathfrak{p}\mathfrak{q}s &\rightsquigarrow \mathsf{case}(\mathfrak{q}s)(\lambda y^{|B|_{\Im R}}.\,\mathsf{in_L}(\mathsf{pair}\,py))\,\mathsf{in_R}\\ &\rightsquigarrow \mathsf{in_R}\,e_2.\end{aligned}$$

Since it is an exception value, we have to show that e_2 properly extends s. This follows by the assumption that $q\ \mathsf{R}^s_{\Im R}\ B$.

$\mathfrak{p}s \rightsquigarrow$ **$\mathsf{in_R}$ e_1 and $\mathfrak{q}s \rightsquigarrow$ $\mathsf{in_L}$ q** In this case (3.3) reduces as follows:

$$\begin{aligned}\mathfrak{merge}_{\Im R}\,\mathfrak{p}\mathfrak{q}s &\rightsquigarrow \mathsf{case}(\mathfrak{q}s)(\lambda\_^{|B|_{\Im R}}.\,\mathsf{in_R}\,e_1)(\lambda e_2^{\mathrm{Ex}}.\,\mathsf{in_R}(\mathsf{merge}\,e_1 e_2))\\ &\rightsquigarrow \mathsf{in_R}\,e_1\end{aligned}$$

Since it is an exception value, we have to show that e_1 properly extends s. This follows by the assumption that $p\ \mathsf{R}^s_{\Im R}\ A$.

$\mathfrak{p}s \rightsquigarrow$ **$\mathsf{in_R}$ e_1 and $\mathfrak{q}s \rightsquigarrow$ $\mathsf{in_R}$ e_2**

$$\begin{aligned}\mathfrak{merge}_{\Im R}\,\mathfrak{p}\mathfrak{q}s &\rightsquigarrow \mathsf{case}(\mathfrak{q}s)(\lambda\_^{|B|_{\Im R}}.\,\mathsf{in_R}\,e_1)(\lambda e_2^{\mathrm{Ex}}.\,\mathsf{in_R}(\mathsf{merge}\,e_1 e_2))\\ &\rightsquigarrow \mathsf{in_R}(\mathsf{merge}\,e_1 e_2)\end{aligned}$$

Since it is an exception value, we have to show that $\mathsf{merge}\,e_1 e_2$ properly extends s. By Property EX, this happens whenever both e_1 and e_2 properly extends s. This is the case by the assumption that $p\ \mathsf{R}^s_{\Im R}\ A$ and $q\ \mathsf{R}^s_{\Im R}\ B$. □

Following Definition 7, for each state s, the monadic realizability relation $\mathfrak{R}^s_{\Im R}$ induces a monadic realization semantics, which realizes HA by Theorem 1. We employ this family of semantics indexed by a state in order to define another one, which does not depend on a state.

Definition 11 (Interactive Realizability Semantics)**.** *We say that the decorated sequent* $\Gamma \Vdash_{\Im R} \mathfrak{r} : A$ *is valid if and only if it is valid with respect to the semantics induced by each* $\mathfrak{R}^s_{\Im R}$ *for every state* s*.*

We shall show how we can realize EM_1 in this semantics.

3.3.3 Realizing the Excluded Middle Axiom

Interactive realizability aims at producing a realizer of the EM_1 axiom, a weakened form of the excluded middle restricted to Σ^0_1 formulas. A generic instance of EM_1 is written as:

$$\mathsf{EM}_1(P, t_1, \ldots, t_k) \equiv (\forall y.\ P(t_1, \ldots, t_k, y)) \vee (\exists y.\ \neg P(t_1, \ldots, t_k, y)).$$

for any $k+1$-ary relation P and arithmetic terms $t_1, \ldots, t_k$. We call *universal* (resp. *existential*) *disjunct* the first (resp. the second) disjunct of $\mathsf{EM}_1(P, t_1, \ldots, t_k)$. For more information on EM_1 see [1].

The main hurdle we have to overcome in order to build a realizer of $\mathsf{EM}_1(P, t_1, \ldots, t_k)$ is that, by the well-known undecidability of the halting problem, there is no total recursive function that can choose which one of the disjuncts holds. Moreover, if the realizer chooses the existential disjunct, it should also be able to provide a witness.

As we said before terms of type State contain knowledge about witnesses of Σ^0_1 formulas. In order to query a state s for a witness n of $\exists y.\ P(\boldsymbol{n}_1, \ldots, \boldsymbol{n}_k, y)$ for some natural numbers $n_1, \ldots, n_k$, we need to extend system T' with the family of term constants:

$$\mathsf{query}_P : \text{State} \to \underbrace{\text{Nat} \to \cdots \to \text{Nat}}_{k} \to \text{Unit} + \text{Nat}.$$

indexed by $P \in \mathcal{R}_{k+1}$ (and implicitly by $k \geq 0$). The value of $\mathsf{query}_P\, s\boldsymbol{n}_1 \cdots \boldsymbol{n}_k$ should be either $*$ if the s contains no information about such an n or a numeral $\boldsymbol{n}$ such that $[\![P]\!](n_1, \ldots, n_k, n)$ is true. More formally we require that query_P satisfies the following syntactic property:

$$\mathsf{query}_P\, s\boldsymbol{n}_1 \cdots \boldsymbol{n}_k \rightsquigarrow \mathsf{in}_\mathrm{R}\, \boldsymbol{n} \text{ entails that } P(\boldsymbol{n}_1, \ldots, \boldsymbol{n}_k, \boldsymbol{n}) \text{ holds} \tag{IR1}$$

for all natural numbers $n_1, \ldots, n_k$. This amounts to require that state do not answer with wrong witnesses and it follows immediately from the intended interpretation if we suitably define $\mathsf{query}_P\, s\boldsymbol{n}_1 \cdots \boldsymbol{n}_k$ using $[\![s]\!](P, (n_1, \ldots, n_k))$.

An interactive realizer $\mathfrak{r}_P$ of $\mathsf{EM}_1(P)$ will behave as follows. When it needs to choose one of the disjuncts it queries the state. If the state answer with a witness, $\mathfrak{r}_P$ reduces to a realizer $\mathfrak{r}_\exists$ of the existential disjunct containing the witness given by the state. Otherwise we can only assume (since we do not know any witness) that the universal disjunct holds and thus $\mathfrak{r}_P$ reduces to a realizer $\mathfrak{r}_\forall$ of the universal disjunct. This assumption may be wrong if the state is not big enough. When $\mathfrak{r}_\forall$ is evaluated on numerals (this correspond to the fact that an instance $P(\boldsymbol{n}_1, \dots, \boldsymbol{n}_k, \boldsymbol{n})$ of the universal disjunct assumption is used in the proof), $\mathfrak{r}_\forall$ checks whether the instance holds. If this is not the case the realizer made a wrong assumption and $\mathfrak{r}_\forall$ reduces to an exceptional value, with the effect of halting the regular reduction and returning the exceptional value. For this we need to extend the system T' with the last family of terms:

$$\mathsf{eval}_P : \underbrace{\mathrm{Nat} \to \cdots \to \mathrm{Nat}}_{k} \to \mathrm{Nat} \to \mathrm{Unit} + \mathrm{Ex},$$

again indexed by $P \in \mathcal{R}_k$. We shall need eval_P to satisfy the following property:

$$\mathsf{eval}_P\, \boldsymbol{n}_1 \cdots \boldsymbol{n}_k \boldsymbol{n} \rightsquigarrow \mathsf{in}_\mathrm{L} * \text{ entails that } P(\boldsymbol{n}_1, \dots, \boldsymbol{n}_k, \boldsymbol{n}) \text{ does not hold,} \tag{IR2}$$

for all natural numbers $n_1, \dots, n_k, n$. This guarantees that if the universal disjunct instance does not hold eval_P reduces to an exceptional value. Thus an interactive realizer which uses a false instance of an universal assumption cannot reduce to a regular value.

The last property we need is that for any state s and natural numbers $n_1, \dots, n_k$,

$$\left.\begin{array}{r} \mathsf{query}_P\, s\boldsymbol{n}_1 \cdots \boldsymbol{n}_k \rightsquigarrow \mathsf{in}_\mathrm{L} * \\ \mathsf{eval}\, \boldsymbol{n}_1 \cdots \boldsymbol{n}_k \rightsquigarrow \mathsf{in}_\mathrm{R}\, e \end{array}\right\} \text{ entails that } e \text{ properly extends } s. \tag{IR3}$$

This condition guarantees that we have no "lazy" realizers that throw exceptions encoding witnesses that are already in the state.

ow we can define a realizer for $\mathsf{EM}_1(P, t_1, \dots, t_k)$ as follows:

$$\begin{aligned} \mathsf{em}_\mathrm{N}(P, t_1, \dots, t_k) \equiv \lambda s^{\mathrm{State}}.\, \mathsf{in}_\mathrm{L}(\mathsf{case}(&\mathsf{query}_P\, st_1 \cdots t_k) \\ &(\lambda\_^{\mathrm{Unit}}.\, \mathsf{in}_\mathrm{L}(\lambda y^{\mathrm{Nat}}.\lambda\_^{\mathrm{State}}.\, \mathsf{eval}_P\, t_1 \cdots t_k y)) \\ &(\lambda y^{\mathrm{Nat}}.\, \mathsf{in}_\mathrm{R}(\mathsf{pair}\, y\, \mathfrak{unit}_{\mathfrak{I}R}))). \end{aligned}$$

Of course we need to check that our definition is correct.

Proposition 5 (Interactive Realizer for EM_1)**.** *Given any* EM_1 *instance* $\mathsf{EM}_1(P, t_1, \ldots, t_k)$*, the decorated sequent:*

$$\alpha_1 : A_1, \ldots, \alpha_l : A_l \Vdash_{\Im R} em_N(P, t_1, \ldots, t_k) : \mathsf{EM}_1(P, t_1, \ldots, t_k), \tag{3.4}$$

is valid with respect to the interactive realizability semantics given in Definition 11.

Proof. Let $\mathfrak{r}$ and A stand for $\mathsf{em}_\mathrm{N}(P, t_1, \ldots, t_k)$ and $\mathsf{EM}_1(P, t_1, \ldots, t_k)$ in the following proof. By Definition 11, we have to prove that (3.4) is valid with respect to the semantics induced by $\mathfrak{R}^s_{\Im R}$ for any given state s.

Let the free (arithmetic) variables of A be $x_1, \ldots, x_m$ and let $\Omega \equiv x_1 \coloneqq \boldsymbol{n}_1, \ldots, x_m \coloneqq \boldsymbol{n}_m$ be a substitution for them. Let Σ be a substitution for the assumption variables in Γ. Note that the only free variables in $\mathfrak{r}$ are arithmetic, thus $\mathfrak{r}[\Sigma]$ is the same as $\mathfrak{r}$.

Thus we have to prove that

$$\mathfrak{r}[\Sigma, \Omega] \; \mathfrak{R}^s_{\Im R} \; A[\Omega].$$

By definition of $\mathfrak{R}^s_{\Im R}$, we apply s and reduce:

$$\begin{aligned} \mathfrak{r}[\Sigma, \Omega] s \rightsquigarrow \mathsf{in}_\mathrm{L}(&\mathsf{case}(\mathsf{query}_P\, s t_1[\Omega] \cdots t_k[\Omega]) \\ &\quad (\lambda \_^{\mathrm{Unit}}.\, \mathsf{in}_\mathrm{L}(\lambda y^{\mathrm{Nat}}.\lambda \_^{\mathrm{State}}.\, \mathsf{eval}_P\, t_1[\Omega] \cdots t_k[\Omega] y)) \\ &\quad (\lambda y^{\mathrm{Nat}}.\, \mathsf{in}_\mathrm{R}(\mathsf{pair}\, y\, \mathfrak{unit}_{\Im R}))), \end{aligned}$$

and since $\mathfrak{r}[\Sigma, \Omega]s$ is a regular value, $\mathfrak{r}[\Sigma, \Omega]$ is a monadic realizer of A if and only if:

$$\begin{aligned} &\mathsf{case}(\mathsf{query}_P\, s t_1[\Omega] \cdots t_k[\Omega]) \\ &\quad (\lambda \_^{\mathrm{Unit}}.\, \mathsf{in}_\mathrm{L}(\lambda y^{\mathrm{Nat}}.\lambda \_^{\mathrm{State}}.\, \mathsf{eval}_P\, t_1[\Omega] \cdots t_k[\Omega] y)) \\ &\quad (\lambda y^{\mathrm{Nat}}.\, \mathsf{in}_\mathrm{R}(\mathsf{pair}\, y\, \mathfrak{unit}_{\Im R})). \end{aligned} \tag{3.5}$$

is an inner realizer for A. $\mathsf{query}_P\, s t_1[\Omega] \cdots t_k[\Omega]$ reduces either to $\mathsf{in}_\mathrm{L} *$ or to $\mathsf{in}_\mathrm{R}\, \boldsymbol{n}$ for some natural number n. We distinguish the two cases.

$\mathsf{in}_\mathrm{L} *$ In the first case (3.5) reduces to:

$$\mathsf{in}_\mathrm{L}(\lambda y^{\mathrm{Nat}}.\lambda \_^{\mathrm{State}}.\, \mathsf{eval}_P\, t_1[\Omega] \cdots t_k[\Omega] y).$$

By definition of $\mathsf{R}^s_{\Im R}$, this is an inner realizer for A if and only if:

$$r_\forall \equiv \lambda y^{\mathrm{Nat}}.\lambda \_^{\mathrm{State}}.\, \mathsf{eval}_P\, t_1[\Omega] \cdots t_k[\Omega] y,$$

is an inner realizer for $\forall y.\ P(t_1[\Omega], \dots, t_k[\Omega], y)$. Again by definition of $\mathsf{R}^s_{\Im R}$, this is the case if and only if

$$r_\forall \boldsymbol{n}\ \Re^s_{\Im R}\ P(t_1[\Omega], \dots, t_k[\Omega], \boldsymbol{n}),$$

for any natural number n. Following the definition of $\Re^s_{\Im R}$, we apply s to $r_\forall \boldsymbol{n}$ and reduce:

$$r_\forall \boldsymbol{n} s \rightsquigarrow \mathsf{eval}_P\, t_1[\Omega] \cdots t_k[\Omega] \boldsymbol{n}$$

Then $r_\forall \boldsymbol{n} s$ reduces either to $\mathsf{in_L} *$ or to $\mathsf{in_R}\ e$, for some exception e.

$\mathsf{in_L} *$ In the first case, we have to check that:

$$* \ \mathsf{R}^s_{\Im R}\ P(t_1[\Omega], \dots, t_k[\Omega], \boldsymbol{n})$$

By definition of $\mathsf{R}^s_{\Im R}$, this is the case if and only if $P(t_1[\Omega], \dots, t_k[\Omega], \boldsymbol{n})$ and this follows from Property IR2.

$\mathsf{in_R}\ e$ In the second case, by definition of $\Re^s_{\Im R}$, we have to check that e properly extends s and this follows from Property IR3.

$\mathsf{in_R}\ \boldsymbol{n}$ In this case, (3.5) reduces to:

$$\mathsf{in_R}(\mathsf{pair}\ \boldsymbol{n}\ \mathrm{unit}_{\Im R}).$$

By definition of $\mathsf{R}^s_{\Im R}$, this is an inner realizer for A if and only if

$$\mathsf{pair}\ \boldsymbol{n}\ \mathrm{unit}_{\Im R}$$

is an inner realizer for

$$\exists y.\ \neg P(t_1[\Omega], \dots, t_k[\Omega], y).$$

Again by definition of $\mathsf{R}^s_{\Im R}$, this is the case if and only if

$$\mathrm{unit}_{\Im R}\ \mathsf{R}^s_{\Im R}\ \neg P(t_1[\Omega], \dots, t_k[\Omega], \boldsymbol{n}).$$

Since $\neg P(t_1[\Omega], \dots, t_k[\Omega], \boldsymbol{n})$ is defined as $P(t_1[\Omega], \dots, t_k[\Omega], \boldsymbol{n}) \to \bot$, again by definition of $\mathsf{R}^s_{\Im R}$, we have to show that:

$$\mathrm{unit}_{\Im R}\ u\ \Re^s_{\Im R}\ \bot,$$

for any inner realizer u of $P(t_1[\Omega], \cdots, t_k[\Omega], \boldsymbol{n})$. However, by Property IR1, $P(t_1[\Omega], \cdots, t_k[\Omega], \boldsymbol{n})$ does not hold, so there is no such u. Thus

$$\mathfrak{unit}_{\mathfrak{I}R}\, u\, \mathfrak{R}^s_{\mathfrak{I}R} \perp$$

holds vacuously.

□

Then we can extend our proof decoration for HA (see Figure 3.1) with the new axiom rule:

$$\mathsf{EM}_1 \; \frac{}{\Gamma \Vdash_{\mathfrak{I}R} \mathsf{em}_{\mathrm{N}}(P, t_1, \ldots, t_k) : \mathsf{EM}_1(P, t_1, \ldots, t_k)}$$

and show that interactive realizability realizes the whole $\mathsf{HA} + \mathsf{EM}_1$.

Theorem 2 (Soundness of $\mathsf{HA} + \mathsf{EM}_1$ with respect to Interactive Realizability Semantics)**.** *Let $\mathcal{D}$ be a derivation of $\Gamma \vdash A$ in $\mathsf{HA} + \mathsf{EM}_1$. Then $\Gamma \Vdash_{\mathfrak{I}R} \mathcal{D}^* : A$, where $\mathcal{D}^*$ is the term obtained by decorating $\mathcal{D}$, is valid with respect to the interactive realizability semantics.*

Proof. By definition of interactive realizability semantics, we have to prove that $\Gamma \Vdash_{\mathfrak{I}R} \mathcal{D}^* : A$ is valid with respect to the monadic realizability semantics induced by $\mathfrak{R}^s_{\mathfrak{I}R}$ for any state s. So we fix a generic state s and proceed by induction on the structure of the decorated version of $\mathcal{D}$, exactly as in Theorem 1, that is, we prove that each rule whose premisses are valid has a valid conclusion. Since $\mathfrak{R}^s_{\mathfrak{I}R}$ is a monadic realizability relation, this has already been shown in the proof of Theorem 1 for all the rules in HA. We only need to check the EM_1 axiom, but we have already done this in Proposition 5. □

3.4 Conclusions

As we mentioned in the introduction, interactive realizability describes a learning by trial-and-error process. In our presentation we focused on the evaluation of interactive realizers, which corresponds to the trial-and-error part and is but a single step in the learning process. For the sake of completeness, we briefly describe the learning process itself.

We can interpret an interactive realizer $\mathfrak{r}$ of a formula A as a function f from states to states. Recall that the intended interpretation of a term e : Ex is a function that extends

states. Then we can define f by means of $\mathfrak{r}$ as follows:

$$f(s) = \begin{cases} [\![e]\!](s) & \text{if } \mathfrak{r} \rightsquigarrow \mathsf{in}_\mathrm{R}\, e, \\ s & \text{if } \mathfrak{r} \rightsquigarrow \mathsf{in}_\mathrm{L}\, t \text{ for some t.} \end{cases}$$

Note that by definition of $\Re_{\Im R}$ we know that in the first case $[\![e]\!]s$ properly extends s. We can think of f as a learning function: we start from a knowledge state and try to prove A with $\mathfrak{r}$. If we fail, we learn some information that was not present in the state and we use it to extend the state. If we succeed then we do not learn anything and we return the input state. Thus note that the fixed points of f are exactly the states containing enough information to prove A.

By composing f with itself we obtain a learning process: we start from some state (for instance the empty one) and we apply f repeatedly. If in this repeated application eventually produces a fixed point, the learning process ends, since we have the required information to prove A. Otherwise we build an infinite sequence of ever increasing knowledge states whose information is never enough to prove A. The fact that the learning process described by interactive realizability ends is proved in Theorem 2.15 of [2].

We wish to point out one of the main differences between our presentation of interactive realizability and the one given in [2]. In [2], the formula-as-types correspondence is closer to the standard one. Exceptions are allowed only at the level of atomic formulas and merge is only used in atomic rules. For instance a realizer for a conjunction $A \wedge B$ could normalize to pair $e_1 e_2$. In this case, the failure of the realizer is not apparent (at least at the top level) and it is not clear which one of e_1 or e_2 we are supposed to extend the state with. In our version exceptions are allowed at the top level of any formula and they "climb" upwards whenever possible by means of merge.

Chapter 4

A Witness Extraction Technique by Proof Normalization

We present a new set of reductions for derivations in natural deduction that can extract witnesses from closed derivations of simply existential formulas in Heyting Arithmetic (HA) plus the law of the excluded middle restricted to simply existential formulas (EM_1).

The reduction we present are inspired by the informal idea of learning by making falsifiable hypothesis and checking them, and by the interactive realizability interpretation. We extract the witnesses directly from derivations in $\mathsf{HA} + \mathsf{EM}_1$ by reduction, without encoding derivations by a realizability interpretation.

4.1 Introduction

In proof theory there are reductions that express the computational interpretation we give to logical connectives, quantifiers and, in the case of arithmetic, induction. Proofs in intuitionistic logic are shown to produce a witness for existential statements: any proof can be reduced to normal form, in which no more reductions are possible, and in a normal proof of an existential statement a witness always appears in a predictable location. We want to obtain the same result for proofs of semi-decidable statements in intuitionistic logic augmented with EM_1 and reduction rules inspired by a trial-and-error interpretation.

We work in Heyting Arithmetic (HA) extended with EM_1, which is weaker than classical

arithmetic but strong enough to prove non-trivial non-constructive results: for instance the fact that every function $f : \mathbb{N} \to \mathbb{N}$ has a minimum. By modifying the standard reductions for Heyting Arithmetic (see [20]), we show that normal proofs of existential statements in $\mathsf{HA} + \mathsf{EM}_1$ produce a witness[1], as they do in the intuitionistic case.

The fact that classical arithmetic is a conservative extension of HA for Π^0_2 statements is well known and the fact that we can extract witnesses from classical proofs of Σ^0_1 statements follows immediately. However proofs of these results usually employ the Gödel-Gentzen negative translation combined with variants of Kreisel's modified realizability semantics or Friedman's translation. Here, by purely proof theoretical means, we prove a slightly weaker result without resorting to negative translations and using reductions justified in terms of Interactive Realizability.

An important remark is that in this chapter we do not prove strong normalization, but a just a result on the form of normal proofs. A formal type theoretic version of the reductions given in the following and strong normalization proof could not be included in this dissertation for reasons of time, but it will appear in [3]. In this section we prove that, *if* we have normalization, then all derivations of simply existential statements compute a witness by a method we describe as trial-and-error.

4.2 A Formal System for Intuitionistic Arithmetic

As usual we work in $\mathsf{HA} + \mathsf{EM}_1$, Heyting Arithmetic extended with the law of the excluded middle for Σ^0_1 formulas. The full description is in Section 2.1.

Since our reduction technique could conceivably be used in other first-order theories, we isolate some general assumptions on atomic formulas and rules that we need for our results to hold:

- closed atomic formulas are decidable,
- any true closed atomic formula has an atomic derivation,
- atomic rules do not discharge assumptions,

[1] Under suitable assumptions on the proof.

- atomic rules do not bind term variables[2].

The first two assumption are very reasonable in a constructive setting such as arithmetic where we expect to have decidability at least for atomic formulas[3]. The other two seems also reasonable for any first-order theory. These assumption are reasonable in a constructive setting and they are satisfied in HA.

We assumed that any true closed atomic formula has an atomic derivation. For convenience we add atomic rules for proving them in one step. Let P be a closed atomic formula. If P is true then we add the atomic axiom:

$$\mathcal{A}\mathrm{I}\ \frac{}{P}$$

Otherwise if P is false we add the atomic rule:

$$\mathcal{A}\mathrm{E}\ \frac{P}{\bot}$$

In order to work on the structure of derivations we need suitable notation and terminology. We represent derivations as upward growing trees of formulas and we make a distinction between a formula (resp. rule) and its occurrences (resp. instances) in a derivation.

A formula can occur more than once in a derivation. While these occurrences are clearly distinct in a tree-like representation, in order to avoid confusion when referring to them in the text special care must be taken. Thus we make a distinction between formulas and *formula occurrences*, or simply occurrences, which we label with $\mathfrak{a}, \mathfrak{b}, \mathfrak{c}$. In a derivation, formula occurrences are arranged following the patterns given by the inference rules. As with formulas, we distinguish between rules and *rule instances*, or simply instances, which we label with α, β, γ. We write a derivation Π as follows:

$$\text{rulename}\ \frac{\dfrac{\Pi_1}{B^{\mathfrak{b}_1}} \quad \dfrac{\begin{matrix}[C]^{\beta}\\ \Pi_2\end{matrix}}{B^{\mathfrak{b}_2}}}{A^{\mathfrak{a}}}\ \alpha$$

The only occurrence of the formula A is labeled $\mathfrak{a}$, while B occurs two times, with distinct labels $\mathfrak{b}_1$ and $\mathfrak{b}_2$. $\mathfrak{a}$ is the *conclusion* of an instance, labeled α, of an inference rule named

[2]The precise meaning of this will be made precise later.

[3]However they may very well fail in set theory, for instance with the inclusion predicate.

rulename. $\mathfrak{a}_1$ and $\mathfrak{a}_2$ are the *premisses* of α. We also say that $\mathfrak{a}$ is the *conclusion* of the whole derivation Π. With Π_1 and Π_2 we denote two *subderivations* (as in subtree) of Π. We distinguish subderivations by their conclusion, so we say that Π_i is the (sub)derivation of $\mathfrak{a}_i$ for $i = 1, 2$. By writing $[C]^\beta$ in square brackets above Π_2, we make explicit that Π_2 may contain occurrences of the assumption C, which is discharged by some undisplayed rule instance β.

We define *assumptions* and *open assumptions* as usual in natural deduction, see [24], page 23.

4.3 The Standard Reductions

In this section we introduce the standard reductions we need for proofs in natural deduction.

Reductive proof theory stems from the following observation: there are derivations that are more complex than they need to be because they have unnecessary detours. When this occurs, we can produce simpler and more direct derivations with the same conclusion by simple structural manipulations called *reductions*.

In standard reductive proof theory for natural deduction, several reductions are introduced: proper reductions, permutative reductions, immediate simplifications and a reduction for the induction rule (see [20]). A derivation is said to be *fully normal* when none of these reductions can be performed on it. For our purposes fully normal derivations are not required, so we introduce only the proper reductions and the induction reduction.

In an instance of an elimination rule, the premiss containing the connective or quantifier that is being eliminated is called the *major* premiss; the other premisses are called the *minor* premisses. We always display the major premiss in the leftmost position.

4.3.1 Proper Reductions

Consider a derivation in which a formula occurrence $\mathfrak{a}$ is both the conclusion of an introduction rule instance α and the major premiss of an elimination rule instance β. Then we can derive the conclusion of β directly by removing α and β and rearranging the derivations of the premisses of α and of the minor premisses of β (if any). Note that α and β must be instances of an introduction rule and an elimination rule for the same logical connective, since

the formula introduced by α is the same formula eliminated by β. Therefore for each logical connective we have a different type of *proper reduction*. They are listed in Figure 4.1.

Figure 4.1: The proper reductions.

$\wedge-$red

$$\wedge\mathrm{E}\,\dfrac{\wedge\mathrm{I}\,\dfrac{\dfrac{\Pi_1}{A_1} \qquad \dfrac{\Pi_2}{A_2}}{A_1 \wedge A_2}\,\alpha}{A_i}\,\beta \quad \xrightarrow{\wedge-\text{red}} \quad \dfrac{\Pi_i}{A_i}$$

$\vee-$red

$$\vee\mathrm{E}\,\dfrac{\vee\mathrm{I}\,\dfrac{\dfrac{\Pi}{A_i}}{A_1 \vee A_2} \qquad \dfrac{\begin{matrix}[A_1]^\alpha \\ \Pi_1\end{matrix}}{B} \qquad \dfrac{\begin{matrix}[A_2]^\alpha \\ \Pi_2\end{matrix}}{B}}{B}\,\alpha \quad \xrightarrow{\vee-\text{red}} \quad \dfrac{\dfrac{\Pi}{A_i}\\ \Pi_i}{B}$$

$\rightarrow-$red

$$\rightarrow\mathrm{E}\,\dfrac{\rightarrow\mathrm{I}\,\dfrac{\dfrac{\begin{matrix}[A]^\alpha \\ \Pi_1\end{matrix}}{B}}{A \rightarrow B}\,\alpha \qquad \dfrac{\Pi_2}{A}}{B} \quad \xrightarrow{\rightarrow-\text{red}} \quad \dfrac{\dfrac{\Pi_2}{A}\\ \Pi_1}{B}$$

$\forall-$red

$$\forall\mathrm{E}\,\dfrac{\forall\mathrm{I}\,\dfrac{\Pi}{\forall x.\, A}}{A[x := t]} \quad \xrightarrow{\forall-\text{red}} \quad \dfrac{\Pi[x := t]}{A[x := t]}$$

$\exists-$red

$$\exists\mathrm{E}\,\dfrac{\exists\mathrm{I}\,\dfrac{\dfrac{\Pi_1}{A[x := t]}}{\exists x.\, A} \qquad \dfrac{\begin{matrix}[A[x := y]]^\alpha \\ \Pi_2\end{matrix}}{B}}{B}\,\alpha \quad \xrightarrow{\exists-\text{red}} \quad \dfrac{\dfrac{\Pi_1}{A[x := t]}\\ \Pi_2[y := t]}{B}$$

4.3.2 Induction Reduction

Consider the induction rule schema Ind in the following form:

$$\text{Ind}\,\dfrac{\dfrac{\Pi_1}{A[x := \mathbf{0}]} \qquad \dfrac{\begin{matrix}[A]^\alpha \\ \Pi_2\end{matrix}}{A[x := \mathsf{succ}(x)]}}{A[x := t]}\,\alpha$$

We call t the *main term* of the induction. An instance α of the Ind rule can be reduced when the main term t in its conclusion $A[x := t]$ is either $\mathbf{0}$ or $\mathsf{succ}(u)$ for some term u. Then if $t = \mathbf{0}$ we can reduce α to:

$$\dfrac{\Pi_1}{A[x := \mathbf{0}]}$$

and if $t = \mathsf{succ}(u)$ as:

$$\dfrac{\dfrac{\dfrac{\Pi_1}{A[x := \mathbf{0}]} \qquad \dfrac{\begin{matrix}[A]^\beta \\ \Pi_2\end{matrix}}{A[x := \mathsf{succ}(x)]}}{A[x := u]}\,\beta \;\text{Ind} \\ \Pi_2[x := u]}{A[x := \mathsf{succ}(u)]}$$

We call this conditional reduction Ind−red. It is easy to see that this reduction is "unraveling" the induction. When u is a numeral $\boldsymbol{n}$, that is, a term of the form succ^n, we can apply the Ind−red reduction repeatedly (n times) until we remove all occurrences of the Ind rule and get:

$$\begin{array}{c} \Pi_1 \\ \hline A[x := \mathbf{0}] \\ \Pi_2[x := \mathbf{0}] \\ \hline A[x := \mathbf{1}] \\ \Pi_2[x := \mathbf{1}] \\ \hline A[x := \mathbf{2}] \\ \vdots \\ A[x := \boldsymbol{n}] \end{array}$$

4.4 The Witness Extracting Reductions

In this section we introduce an inference rule that is equivalent to the restricted excluded middle axiom schema EM_1 defined in Definition 3 and two reductions involving this new

rule. The first one, the Wit−red reduction, is inspired by Interactive Realizability and it will be instrumental in converting classical derivations into constructive ones. The second one is a permutative reduction and is needed later for technical reasons.

4.4.1 The EM_1 Rule

For convenience we replace the EM_1 axiom schema with the equivalent EM_1 rule:

$$\mathsf{EM}_1\ \dfrac{\begin{matrix}[\forall x.\, P]^{\alpha}\\ \vdots\\ A\end{matrix}\qquad\qquad \begin{matrix}[\neg P[x := y]]^{\alpha}\\ \vdots\\ A\end{matrix}}{A}\ \alpha$$

where the variable y does not occur in A nor in any open assumption that A depends on except occurrences of the assumption $\neg P[x := y]$ (as in the $\exists$E rule).

The EM_1 rule is derived by an $\vee$E rule instance, whose major premiss is an instance of the EM_1 axiom and whose rightmost assumption is the major premiss of an $\exists$E instance:

$$\vee\mathrm{E}\ \dfrac{\mathsf{EM}_1\ \dfrac{}{(\forall x.\, A) \vee (\exists x.\, \neg A)} \qquad \begin{matrix}[\forall x.\, A]^{\alpha}\\ \vdots\\ C\end{matrix} \qquad \exists\mathrm{E}\ \dfrac{[\exists x.\, \neg A]^{\alpha} \qquad \begin{matrix}[\neg A[x := y]]^{\beta}\\ \vdots\\ C\end{matrix}}{C}\ \beta}{C}\ \alpha$$

On the other hand, the EM_1 axiom can be derived from the EM_1 rule by two $\vee$I instances:

$$\mathsf{EM}_1\ \dfrac{\vee\mathrm{I}\ \dfrac{[\forall x.\, A]^{\alpha}}{(\forall x.\, A) \vee (\exists x.\, \neg A)} \qquad \vee\mathrm{I}\ \dfrac{\exists\mathrm{I}\ \dfrac{[\neg A[x := y]]^{\alpha}}{\exists x.\, \neg A}}{(\forall x.\, A) \vee (\exists x.\, \neg A)}}{(\forall x.\, A) \vee (\exists x.\, \neg A)}\ \alpha$$

In the following we refer to the assumption $\forall x.\, P$ in the derivation of the leftmost premiss of the EM_1 rule as the *universal assumption* and to the assumption $\neg P[x := y]$ in the derivation of the rightmost premiss as the *existential assumption*.

We can also write the EM_1 rule in sequent style as:

$$\mathsf{EM}_1\ \dfrac{\Gamma, \alpha : \forall x.\, P \vdash A \qquad \Gamma, \alpha : \exists x.\, \neg P \vdash A}{\Gamma \vdash A}\ \alpha$$

The universal assumption $\forall x.\, P$ is a Π^0_1 formula and thus *negatively decidable*, meaning that a finite piece of evidence is enough to prove it false: a counterexample, a natural

number m such that $P[x := \boldsymbol{m}]$ does not hold. Moreover, if we know that it is false, then a counterexample exists and we can find it in a finite time, in the worst case by means of a blind search through all the natural numbers.

On the other hand, in order to prove the universal assumption, we need a possibly infinite evidence, namely, we may need to check $P[x := \boldsymbol{m}]$ for all natural numbers m and this cannot be effectively done (at least when we have no information on P).

The existential assumption $\neg P[x := y]$ is not actually a existential formula. However it is easy to see that it takes the place of the assumption discharged by the $\exists$E rule.

We say that we can prove the existential assumption true by showing a witness, namely a number m such that $\neg P[x := \boldsymbol{m}]$. Thus the existential assumption behaves as if it were *positively decidable*: when it is true, we have a terminating algorithm to find the finite evidence needed to prove it. However, when it false, we have no way to effectively decide if it is false.

Note that a counterexample m for the universal assumption $\forall x.\ P$ is a witness for the existential assumption since in that case $\neg P[x := \boldsymbol{m}]$ holds.

4.4.2 Witness Reduction

Consider a derivation Π ending with an instance α of the EM_1 rule for the atomic formula P:

$$\mathsf{EM}_1\,\frac{\dfrac{\begin{matrix}[\forall x.\ P]^{\alpha}\\ \Pi_1\end{matrix}}{A} \qquad \dfrac{\begin{matrix}[\neg P[x := y]]^{\alpha}\\ \Pi_2\end{matrix}}{A}}{A}\,\alpha$$

A priori we do not know any counterexample to the universal assumption (we do not even know whether it holds or not), so we begin by looking at how the assumption is used in Π_1. In Π_1, consider all the instances $\beta_1, \ldots, \beta_n$ of the $\forall$E rule whose premiss is an occurrence of the universal assumption $\forall x.\ P$ and whose conclusion is the occurrence of a closed (atomic) formula:

$$\forall\mathrm{E}\,\frac{[\forall x.\ P]^{\alpha}}{P[x := t_1]}\,\beta_1 \qquad \ldots \qquad \forall\mathrm{E}\,\frac{[\forall x.\ P]^{\alpha}}{P[x := t_n]}\,\beta_n$$
$$\vdots \qquad\qquad\qquad\qquad\qquad \vdots$$

These represent the concrete instances of the universal assumption that are used to derive A in Π_1. Since the conclusions of $\beta_1, \ldots, \beta_n$ are closed atomic formulas they are decidable. Therefore we can derive the true concrete instances directly with the atomic axiom $\mathcal{A}$I instead of deducing them from the universal assumption. We distinguish two cases.

- If $P[x := t_i]$ is true for all i we replace each β_i with the atomic axiom for $P[x := t_i]$:

$$\forall\mathrm{E}\,\dfrac{[\forall x.\, P]^{\alpha}}{\begin{array}{c}P[x := t_i]\\ \vdots\end{array}}\,\beta_i \quad\rightsquigarrow\quad \mathcal{A}\mathrm{I}\,\dfrac{}{\begin{array}{c}P[x := t_i]\\ \vdots\end{array}}$$

  We call this new derivation Π_1'.

  Now two situations are possible: either Π_1 needs the universal assumption only to deduce the concrete instances $\beta_1, \ldots, \beta_n$ or not.

  – The first case happens when Π_1' contains no more occurrences of the universal assumption discharged by α, that is, the universal assumption only occurs in Π_1 as the premiss of $\beta_1, \ldots, \beta_n$. In this case Π_1' is a self-contained derivation of A and we can replace the whole Π with Π_1'.

  – Otherwise Π_1' still contains some occurrence of the universal assumption. Then Π_1' does need the universal assumption itself and not just some concrete instances of it. In this case we can only replace Π_1 with Π_1' in Π, but we cannot eliminate the EM_1 rule instance α from the derivation.

- Otherwise there is some i such that $P[x := t_i]$ is false. Thus the universal assumption itself is false, since we have found the counterexample t_i. Moreover t_i is a witness for the existential assumption, meaning that we can replace y with t_i in Π_2 and all the occurrences of the assumption $\neg P[x := y]$ with a derivation of $\neg P[x := t]$:

$$\begin{array}{c}[\neg P[x := t_i]]^{\alpha}\\ \vdots\end{array} \quad\rightsquigarrow\quad \rightarrow\mathrm{I}\,\dfrac{\mathcal{A}\mathrm{E}\,\dfrac{[P[x := t_i]]^{\beta_i'}}{\bot}}{\begin{array}{c}\neg P[x := t_i]\\ \vdots\end{array}}\,\beta_i'$$

  We call this new derivation Π_2'.

  Note that in this case we replace all the occurrences of the existential assumption in Π_2 and thus Π_2' is self-contained derivation of A. Therefore we can replace Π with Π_2'.

We call this reduction Wit−red.

The gist of the Wit−red reduction is that we look for counterexamples to the universal assumption in Π_1. If we do not find one then we have checked that all the concrete instances of the universal assumption hold. Moreover if Π_1 uses the universal assumption exclusively

to deduce these concrete instances, then we get a direct derivation of A without using the EM_1 rule. On the other hand if we find a counterexample then we know that we can put it in Π_2 and get another direct derivation of A.

In some sense we have a procedure to decide which one of the subderivation of the EM_1 rule is the effective one, Note that this procedure fails when we do not find counterexamples to the universal assumption but we cannot completely eliminate its occurrences from Π_1. Our main result can be thought of as the proof that, when the conclusion of a derivation is simply existential, this “failure” of the procedure does not happen. The whole reduction is summarized in Figure 4.2.

4.4.3 Permutative Reduction for EM_1

The permutative reduction for EM_1 is defined in the same way as the permutative reduction for the $\vee$E rule, that is, when the conclusion of a EM_1 rule instance is the major premiss of an elimination rule instance $*$E:

$$*\mathrm{E}\;\dfrac{\mathsf{EM}_1\;\dfrac{\begin{array}{c}[\forall x.\, P]^{\gamma}\\ \underline{\;\Pi_1\;}\\ A\end{array} \qquad \begin{array}{c}[\neg P[x := y]]^{\gamma}\\ \underline{\;\Pi_2\;}\\ A\end{array}}{A}\,\gamma \qquad \bar{\Pi}}{B}$$

reduces to:

$$\mathsf{EM}_1\;\dfrac{*\mathrm{E}\;\dfrac{\begin{array}{c}[\forall x.\, P]^{\alpha}\\ \underline{\;\Pi_1\;}\\ A\end{array} \qquad \bar{\Pi}}{B} \qquad *\mathrm{E}\;\dfrac{\begin{array}{c}[\neg P[x := y]]^{\alpha}\\ \underline{\;\Pi_2\;}\\ A\end{array} \qquad \bar{\Pi}}{B}}{B}\,\alpha$$

where $\bar{\Pi}$ stands for the derivations of the remaining minor premisses of β if any. We denote this reduction as EM_1−perm. More explicitly, we can define a permutative reduction for each elimination rule, see Figure 4.3 and Figure 4.4.

This reduction moves elimination rule instances from “outside” or “below” to “inside” or “above” an EM_1 rule instance. This is useful because an EM_1 rule instance may happen in between an introduction rule instance and an elimination rule instance, preventing a proper reduction from taking place.

In the following we concentrate on proving a result about the form of normal proofs. We do not prove here that the reduction process converges. In order to do it, the most natural

way would be encode our proofs into proof terms in a suitable calculus and show that such calculus is strongly normalizing. This has been done in [3], so we just state the following

Theorem 3 (Strong Normalization of $\mathsf{HA} + \mathsf{EM}_1$)**.** *All proofs of* $\mathsf{HA} + \mathsf{EM}_1$ *are strongly normalizing under the reductions we described in this section.*

A proof can be found in [3].

4.5 Witness Extraction

In this section we prove the witness extraction theorem, that shows how we can extract witnesses from suitable classical derivations in $\mathsf{HA} + \mathsf{EM}_1$, as we can do for intuitionistic derivations in HA.

In order to state and prove our results we need to keep track of free term variables in a derivations, since both the Ind−red and the Wit−red reductions can only be performed when certain terms and formulas are closed.

We need to define when a variable is free in a derivation.

Definition 12 (Free term variables)**.** *We say that a rule instance α* binds *a term variable that occurs free in the derivation Π of a premiss of α in the following cases:*

- *α is an instance of the $\forall$I rule and binds the variable x in the formula occurrences in the derivation of its premiss:*

$$\forall\mathrm{I}\,\frac{\dfrac{\Pi}{A}}{\forall x.\, A}\,\alpha$$

- *α is an instance of the $\exists$E rule and binds the variable y in the formula occurrences in the derivation of its rightmost premiss:*

$$\exists\mathrm{E}\,\frac{\exists x.\, A \qquad \begin{matrix}[A[x := y]]^{\alpha} \\ \dfrac{\Pi}{B}\end{matrix}}{B}\,\alpha$$

- *α is an instance of the* Ind *rule and binds the variable x in the formula occurrences in the derivation of its rightmost premiss:*

$$\text{Ind}\ \dfrac{A[x := \mathbf{0}] \qquad \dfrac{\begin{matrix}[A]^{\alpha} \\ \Pi\end{matrix}}{A[x := \mathsf{succ}(x)]}}{A[x := t]}\ \alpha$$

- *α is an instance of the* EM_1 *rule and binds the variable y in the formula occurrences in the derivation of its rightmost premiss:*

$$\mathsf{EM}_1\ \dfrac{\begin{matrix}[\forall x.\, P]^{\alpha} \\ \vdots \\ B\end{matrix} \qquad \begin{matrix}[\neg P[x := y]]^{\alpha} \\ \dfrac{\Pi}{B}\end{matrix}}{B}\ \alpha$$

We say that a term variable occurrence is free in a derivation *when the term variable occurs free in a formula occurrence in the derivation and is not bound by any rule instance. A derivation is* closed *if it has no free term variable nor open assumption.*

Note that no reduction introduces free term variables in a derivation.

Since a derivation is a tree, it makes sense to give the definition of branch. Principal branches are branches of a derivation that contains only major premisses of elimination and EM_1 rule instances.

Definition 13 (Principal branch)**.** *A* branch *in a derivation Π is a sequence of formula occurrences $\mathfrak{a}_0, \dots, \mathfrak{a}_n$ in Π such that:*

- *$\mathfrak{a}_0$ is a top formula occurrence, that is, $\mathfrak{a}_0$ is either an assumption or the conclusion of an atomic axiom;*

- *$\mathfrak{a}_i$ and $\mathfrak{a}_{i+1}$ are respectively a premiss and the conclusion of the same rule instance α_{i+1}, for all $0 \le i < n$;*

- *$\mathfrak{a}_n$ is the conclusion of* Π.

A branch is principal *if, for all $0 \le i < n$ such that α_i is an elimination or* EM_1 *rule instance, $\mathfrak{a}_i$ is the major (leftmost) premiss of α_i.*

We use the variables ζ, η for branches.

In order to study the properties of normal proofs we only need to consider the structure of principal branches. A head-cut is the lowest point of a principal branch where a reduction is possible.

Definition 14 (Head-cut). *The* head-cut *of a principal branch $\zeta = \mathfrak{a}_1, \ldots, \mathfrak{a}_n$ is the formula occurrence $\mathfrak{a}_i$ with the maximum index i such that one of the following holds:*

- *$\mathfrak{a}_i$ is the conclusion of an elimination rule instance α_i, $\mathfrak{a}_{i-1}$ is the major premiss of α_i and the conclusion of an introduction rule instance α_{i-1}; when α_{i-1} is a $\wedge$I rule instance we also require that $\mathfrak{a}_{i-2}$ is an occurrence of the same formula as $\mathfrak{a}_i$ (proper reductions);*
- *$\mathfrak{a}_i$ is the conclusion of an* Ind *rule instance α_i, whose main term is either $\mathbf{0}$ or $\mathsf{succ}(u)$ for some term u (*Ind−red *reduction);*
- *$\mathfrak{a}_i$ is the conclusion of an EM_1 rule instance α and either $\mathfrak{a}_{i-1}$ is derived without using the assumption discharged by α or $\mathfrak{a}_0$ is an occurrence of the universal assumption discharged by α and $\mathfrak{a}_1$ is the occurrence of a closed atomic formula (Wit−*red *reduction);*
- *$\mathfrak{a}_i$ is the conclusion of an elimination rule instance α and $\mathfrak{a}_{i-1}$ is the conclusion of an EM_1 rule instance (*EM_1−perm *reductions).*

If such an i exists we say that there is a head-cut along the branch ζ.

This definition is the result of a analysis of the conditions that must be met in order to perform one of the reductions we have listed. In particular note how, in the condition given for the Wit−red reduction, the fact that $\mathfrak{a}_1$ is atomic implies that $\mathfrak{a}_1$ is the conclusion of a $\forall$E rule instance, as we assumed in defining Wit−red.

We shall show that, with suitable assumptions, we can perform the Wit−red reduction as needed in order to extract a witness from a derivation. One of these assumptions is that the conclusion of the derivation is “simple” enough, as we define next.

Definition 15 (Simple Formulas). *We say that a formula is* simply existential *(resp.* universal*) when it is $\exists x.\ P$ (resp. $\forall x.\ P$) for some atomic formula P.*

We say that a formula is simple *when it is closed and atomic or simply existential.*

In the following we consider the EM_1 and Ind rules to be neither elimination nor introduction rules and we give them special treatment.

As we shall show later, principal branches beginning with an open assumption have particular structure in normal derivations: they begin with a sequence of elimination rule instances, followed by atomic and EM_1 rule instances and they end with introduction and EM_1 rule instances. Any of these parts may be missing.

Definition 16 (Open normal form)**.** *A principal branch* $\mathfrak{a}_0, \dots, \mathfrak{a}_n$ *is said to be in* open normal form *when there exist three natural numbers* n_E, n_A *and* n_I *such that* $n_E + n_A n_I = n$ *and:*

- $\mathfrak{a}_0$ *is the occurrence of an open assumption in* Π,
- $\mathfrak{a}_i$ *is the conclusion of an elimination rule instance for* $0 < i \leq n_E$,
- $\mathfrak{a}_i$ *is the conclusion of an atomic or* EM_1 *rule instance for* $n_E < i \leq n_E + n_A$,
- $\mathfrak{a}_{n_E+n_A+1}$ *is the conclusion of an introduction rule instance*[4],
- $\mathfrak{a}_i$ *is the conclusion of an introduction or* EM_1 *rule instance for* $n_E + n_A < i \leq n$,

n_E, n_A *and* n_I *are the number of elimination, atomic or* EM_1, *introduction or* EM_1 *rule instances, respectively.*

We can now prove our main result: closed normal derivations of simply existential formulas in $\mathsf{HA} + \mathsf{EM}_1$ can be reduced to derivations ending with an introduction rule instance. Derivations in HA have a similar property. The theorem we are going to prove holds for derivations that are concrete enough, namely they are: self-contained (without open assumptions), concrete (without open term variables) and with an effective conclusion (a simply existential formula). The proof is split into several lemmas.

In the first lemma we show that, in a derivation of a simply existential with no free term variables, a simply universal assumption is followed by a closed atomic formula. This will be used later to prove that we can perform the Wit−red reduction on universal assumption of an EM_1 rule instance.

[4] Since EM_1 rule instances can appear intermingled with both atomic and introduction rule instances, in the definition we require that $\mathfrak{a}_{n_E+n_A+1}$ be the conclusion of an introduction rule, so that n_A and n_I are uniquely determined.

Lemma 1. *Let $\zeta = \mathfrak{a}_0, \dots, \mathfrak{a}_n$ be a principal branch in open normal form in a derivation Π in $\mathsf{HA} + \mathsf{EM}_1$, with n_E, n_A and n_I defined as in Definition 16. Let $A_0, \dots, A_n$ be the formulas $\mathfrak{a}_0, \dots, \mathfrak{a}_n$ are occurrences of. Then the following statements hold:*

1. *A_i is a non-atomic subformula of A_n for all $n_E + n_A < i \leq n$;*
2. *if some $\mathfrak{a}_i$ is the conclusion of an introduction rule instance, then A_i is a subformula of A_n;*

Moreover assume that A_n is a simple formula. Then:

3. *if a term variable x is free in some A_i, then x is free in Π;*
4. *if some A_i is simply universal, then $\mathfrak{a}_i$ is the premiss of a $\forall$E rule instance;*
5. *if Π has no free term variables and A_i is simply universal, then A_{i+1} is a closed atomic formula.*

Proof. (2) follows immediately from (1). We need (2) to prove (3) and (4). Then, by (3) and (4), we prove (5). Here are the proofs.

1. We proceed by induction on n_I.
   - If $n_I = 0$, the thesis holds vacuously.
   - If $n_I = 1$, we need to prove the statement just for $i = n_E + n_A + 1 = n$ and thus $\mathfrak{a}_n$ is the conclusion of an introduction rule instance by Definition 16. This means that A_n is not atomic. Obviously it is also a subformula of itself so we are done.
   - Otherwise, let $n_I > 1$.

     Consider the subderivation Π' of Π ending with $\mathfrak{a}_{n-1}$ and its principal branch $\zeta' = \mathfrak{a}_0, \dots, \mathfrak{a}_{n-1}$. ζ' is in open normal form in Π', with $n'_E = n_E, n'_A = n_A$ and $n'_I = n_I - 1$. Then, by inductive hypothesis, for all $n_E + n_A < i \leq n - 1$, A_i is a non-atomic subformula of A_{n-1}.

     By Definition 16, Π ends with an introduction or EM_1 rule instance α. In both cases A_{n-1} is a subformula of A_n, since $\mathfrak{a}_{n-1}$ is the premiss of α and $\mathfrak{a}_n$ is its conclusion.

     Thus for all $n_E + n_A < i \leq n$, A_i is a subformula of A_n. Moreover since A_{n-1} is non-atomic then A_n is too.

2. If $\mathfrak{a}_i$ is the conclusion of an introduction rule instance then $n_E + n_A < i \leq n$ by Definition 16. Thus we conclude by (1).

3. We show that x is not bound by any rule instance and thus is free in Π. The only rule that binds a variable above a major premiss, and thus the only rule that can bind a variable in a principal branch, is the $\forall$I rule. Now assume that a $\forall$I rule instance occur along ζ with conclusion $\mathfrak{a}_j$. By the previous statement (2), A_j is a subformula of A_n. This yields a contradiction because by assumption A_n is simple and A_j is universally quantified since $\mathfrak{a}_j$ is the conclusion of a $\forall$I rule instance.

4. We show that $\mathfrak{a}_i$ is the premiss of a $\forall$E rule instance because no other alternative is possible.

   - $\mathfrak{a}_i$ cannot be the premiss of an atomic rule instance, since we assumed that A_i is simply universal and thus not atomic.
   - $\mathfrak{a}_i$ cannot be the premiss of an introduction rule instance, since in that case A_{i+1} is a subformula of A_n by (2). Therefore a simply universal formula A_i is a subformula of a simple formula A_n, which is a contradiction.
   - Finally $\mathfrak{a}_i$ cannot be the premiss of an EM_1 rule instance. More precisely assume that $\mathfrak{a}_i$ is followed by exactly $j > 0$ instances of the EM_1 rule. Then $\mathfrak{a}_{i+j}$ is the conclusion of the last EM_1 rule instance α and A_{i+j} is the same formula as A_i, in particular A_{i+j} is simply universal. By definition of open normal form, EM_1 rule instances can only be followed by introduction, atomic or EM_1 rule instances. Since we assumed that there are exactly j instances of the EM_1 rule, $\mathfrak{a}_{i+j}$ is the premiss of either an atomic or introduction rule instance. Then we are in one of the previous cases and we have a contradiction.

   Then $\mathfrak{a}_i$ can only be the premiss of an elimination rule instance and, being A_i simply universal, it must be an instance of the $\forall$E rule.

5. By (4) we known that $\mathfrak{a}_{i+1}$ is the conclusion of a $\forall$E rule instance whose premiss is simply universal. Therefore A_{i+1} is an atomic formula. If A_{i+1} has a free term variable, Π has too by (3). Since we assumed that Π has no free term variable, A_{i+1} must be closed. Therefore A_{i+1} is a closed atomic formula. $\square$

□

In the following lemma we show how we can apply the Wit−red reduction.

Lemma 2 (EM_1 reduction)**.** *Let* Π *be a derivation in* $\mathsf{HA}+\mathsf{EM}_1$ *with no free term variables. Assume that* Π *ends with an* EM_1 *rule instance* α *whose conclusion is an occurrence of a simple formula. Assume that the derivation* Π' *of the leftmost premiss of* α *has a principal branch* ζ *in open normal form. Then at least one of the following occurs:*

1. Π *has a head-cut or a non-normal term along a principal branch,*

2. Π *has a principal branch in open normal form.*

Proof. Let $\zeta = \mathfrak{a}_0, \dots, \mathfrak{a}_n$ and let $A_0, \dots, A_n$ be the formulas $\mathfrak{a}_0, \dots, \mathfrak{a}_n$ are occurrences of. Let $\mathfrak{a}$ be the conclusion of Π and of the EM_1 rule instance α:

$$\mathsf{EM}_1\ \dfrac{\begin{matrix}[A_0^{\mathfrak{a}_0}]\\ \zeta\\ \Pi'\\ A_n^{\mathfrak{a}_n}\end{matrix} \qquad \begin{matrix}\vdots\\ A_n\end{matrix}}{A_n^{\mathfrak{a}}}\ \alpha$$

Note that we can extend ζ to $\eta = \mathfrak{a}_0, \dots, \mathfrak{a}_n, \mathfrak{a}$ and η is a principal branch of Π.

If $\mathfrak{a}_0$ is not discharged by α, then η is a principal branch in open normal form and thus we get the statement. Otherwise, $\mathfrak{a}_0$ is discharged by α, meaning that A_0 is the universal assumption of the EM_1 instance. We can apply (5) of Lemma 1 to η, since $\mathfrak{a}_0$ is simply universal, $\mathfrak{a}$ is simply existential and Π has no free term variables.

Then $\mathfrak{a}_1$ is a closed atomic formula and we can perform the Wit−red reduction, that is, there is a head-cut along the principal branch η of Π and we can conclude. □

□

The following lemma shows how to handle Ind rule instances.

Lemma 3 (Induction normalization)**.** *Let* Π *be a derivation in* $\mathsf{HA}+\mathsf{EM}_1$ *ending with an* Ind *rule instance. Then at least one of the following holds:*

1. Π *has a head-cut or a non-normal term along a principal branch,*

2. Π *contains a free term variable.*

Proof. Let α be the Ind rule instance Π ends with and let its conclusion be an occurrence $\mathfrak{a}$ of some formula $A[x := t]$. If $\mathfrak{a}$ has a free term variable x then x is free in Π too and we are done. If t is not normal then all principal branches[5] in Π have a non-normal term. Otherwise t is a closed normal term and thus it is either $\mathbf{0}$ or $\mathsf{succ}(u)$ for some term u and e we can apply the Ind−red reduction, meaning that any principal branch of Π has a head cut. □ □

The following lemma can be thought of as a weak result on the structure of derivations.

Lemma 4 (Structure of Normal Form)**.** *Let* Π *be a derivation in* $\mathsf{HA} + \mathsf{EM}_1$. *Then at least one of the following holds:*

1. Π *has a head-cut or a non-normal term along a principal branch;*
2. Π *contains a free term variable;*
3. Π *has a principal branch in open normal form;*
4. Π *ends with an introduction rule instance;*
5. Π *is atomic (only atomic formulas occur in* Π*);*
6. Π *ends with an* EM_1 *instance and its conclusion is not simple.*

Proof. The proof is by induction on the structure of the derivation Π, that is, we assume that the statement holds for all subderivations of Π and we prove that it holds for the whole derivation.

Let α be the last rule instance in Π. If α in an introduction rule instance the statement is satisfied and we are done.

If α is an Ind rule instance then we get the statement by applying Lemma 3 to Π.

Then we only need to understand what happens when α is an elimination, an atomic or an EM_1 rule instance. Note that the only case in which α has no premisses is when α is an atomic axiom. If this happens then Π is atomic (it is just the conclusion of α) and the statement is satisfied.

[5] Since all branches of Π end with its conclusion.

Otherwise α has one or more (when α is atomic) major premisses. Let Π' be the derivation of any one of the major premisses of α. Any principal branch ζ of Π' can be extended to a branch η of Π, by appending the conclusion of Π. η is principal too because Π' is the subderivation of a major premiss of α. We shall use this fact often in the following.

By inductive hypothesis Π' satisfies the statement, so we proceed by considering all the possible cases.

1. Π' has a head-cut or a non-normal term along a principal branch ζ. As we noted ζ can be extended to a principal branch of Π with the same head-cut or non-normal term, so Π satisfies the statement and we are done.

2. Π' contains a free term variable.

   There are four rules that can bind term variables: the $\forall$I, $\exists$E, Ind and EM_1 rules. Since the cases when α is an introduction or Ind rule instance have been taken care of already and since the $\exists$E and EM_1 rules can only bind term variables in the derivation of its minor premiss, any free term variable in Π' is free in Π too. Thus Π satisfies the statement.

3. Π' has a principal branch $\zeta = \mathfrak{a}_0, \ldots, \mathfrak{a}_n$ in open normal form. Let η be the principal branch of Π extending ζ.

$$
\begin{array}{c}
[A_0^{\mathfrak{a}_0}] \\
\zeta \\
\Pi' \\
\mathsf{EM}_1 | * \mathrm{E} | \mathcal{A}\ \dfrac{A_n^{\mathfrak{a}_n} \qquad \cdots}{A}\ \alpha
\end{array}
$$

   Note that elimination rule instances do not discharge assumptions in their leftmost subderivation and atomic rule instances do not discharge assumptions in general.

   Then, when α is either an elimination or atomic rule instance, the assumption $\mathfrak{a}_0$ is still open in Π and we have the following cases depending on how which rule $\mathfrak{a}_n$ is the conclusion of. Note that since ζ is in open normal form, $\mathfrak{a}_n$ cannot be an Ind instance. Thus we have the following cases:

		$\mathfrak{a}_n$ is the conclusion of			
		ELIM	ATOM	INTRO	EM_1
α	ELIM	EXT	NO	CUT	PERM
	ATOM	EXT	EXT	NO	NO/EXT

EXT η begins with the open assumption $\mathfrak{a}_0$ followed by elimination and atomic rule instances, so Π satisfies the statement;

NO this is never the case since, in a principal branch, an elimination (resp. atomic) rule instance cannot follow an atomic (resp. introduction) rule instance, because the major premiss (resp. conclusion) of an elimination (resp. introduction) rule instance is not atomic and thus cannot be the conclusion (resp. premiss) of an atomic rule instance;

CUT α is an elimination rule instance and its major premiss is the conclusion of an introduction rule instance, thus η ends with a head-cut and again Π satisfies the statement;

PERM when a major premiss of α is the conclusion of an EM_1 rule instance we can apply the EM_1−perm reduction, thus η ends with a head-cut and Π satisfies the statement;

NO/EXT we have two cases depending on the n_I of ζ:

$n_I > 0$ then, by (1) of Lemma 1, we have that $\mathfrak{a}_n$ is an occurrence of a non atomic formula and thus α cannot be an atomic rule instance;

$n_I = 0$ in this case η is in open normal form and thus Π satisfies the statement.

On the other hand, if α is an EM_1 rule instance then we can apply Lemma 2 to Π (since we can safely assume that Π contains no free term variables) and we get the statement.

4. Π' ends with an introduction rule instance β. Then the conclusion $\mathfrak{b}$ of β cannot be atomic and since it is a premiss of α, α cannot be atomic either. Therefore α must be either an elimination or an EM_1 rule instance.

   If α is an elimination then there is a head-cut along a principal branch going through

$\mathfrak{b}$ so Π satisfies the statement.

$$\dfrac{\dfrac{\vdots}{B^{\mathfrak{b}}}\,?\mathrm{I}\ \beta \qquad \cdots}{A^{\mathfrak{a}}}\,?\mathrm{E}\ \alpha$$

If α is an EM_1 rule instance then $\mathfrak{a}$ and $\mathfrak{b}$ are both occurrences of the formula A. If A is not simple then Π satisfies the statement. Otherwise we assume that A is simple: since A cannot be atomic (it occurs as the conclusion of the introduction rule instance β) A must be $\exists x.\ P$ for some atomic formula P and β must be an $\exists\mathrm{I}$ rule instance. Then we are in the following situation:

$$\dfrac{\dfrac{\dfrac{\Pi''}{A}}{\exists x.\ P^{\mathfrak{b}}}\,\exists\mathrm{I}\ \beta \qquad \dfrac{\vdots}{\exists x.\ P}}{\exists x.\ P^{\mathfrak{a}}}\,\mathsf{EM}_1\ \alpha$$

where Π'' is the derivation of the premiss of β. Again note that principal branches of Π'' extend to principal branches of Π' by appending $\mathfrak{b}$.

By inductive hypothesis Π'' satisfies the statement, so we proceed by considering all the possible cases.

(a) Π'' has a head-cut or a non-normal term along a principal branch. Then Π' does too and we are in the previously solved case labeled 1.

(b) Π'' contains a free term variable. Since $\exists\mathrm{I}$ does not bind free term variables, Π' does too and we are in the previously solved case labeled 2.

(c) Π'' has a principal branch ζ in open normal form. Since $\exists\mathrm{I}$ does not discharge open assumptions, Π' does too and we are in the previously solved case labeled 3.

(d) Π'' ends with an introduction rule instance. This cannot happen because $\mathfrak{b}$ is an atomic formula occurrence.

(e) Π'' is atomic. The assumption discharged by α from its leftmost subderivation is not atomic, thus it cannot occur in Π'' since Π'' is atomic. Therefore we can apply the Wit−red reduction meaning that there is a head-cut at the end of the principal branches of Π' and we are in the previously solved case labeled 1.

(f) Π'' ends with an EM_1 instance and its conclusion is not simple. This cannot happen because we assumed that A is simple and $\mathfrak{b}$ is an occurrence of A.

5. Π' is atomic. In this case one of major premisses of α is atomic, so α cannot be an elimination rule instance and must be either an EM_1 or atomic rule instance.

   - If α is an EM_1 rule instance then it is redundant: the assumption discharged by α from its leftmost subderivation is not atomic, thus it cannot occur in Π' since Π' is atomic whose premisses are atomic formula occurrences. Therefore we can apply the Wit−red reduction to α, meaning that there is a head-cut at the end of the principal branches of Π and thus Π satisfies the statement.
   - Otherwise, if α is an atomic rule instance, consider the other subderivations of its major premisses. If they are all atomic then Π is atomic too and it satisfies the statement. Otherwise there is a major premiss of α with a non atomic derivation Π''. Then one of the other cases applies with Π'' in place of Π'.

6. Π' ends with an EM_1 rule instance β and its conclusion is not simple. α cannot be an atomic rule instance since one of its premisses is the conclusion of β which is not simple and thus not atomic. If α is an elimination rule instance we can apply the EM_1−perm reduction to α and β. Thus there is a head-cut at the end of the principal branches of Π, and Π satisfies the statement. Otherwise, if α is an EM_1 rule instance then the conclusions of Π and Π' are occurrences of the same non-simple formula. Therefore Π again satisfies the statement.

Since we exhausted all the possible cases we are done. □

Our main theorem is now an easy corollary of the previous lemma.

Theorem 4 (Witness Extraction)**.** *Let* Π *be a derivation of a simple formula* A *in* $\mathsf{HA} + \mathsf{EM}_1$. *Assume that:*

1. Π *has no principal branch with a head-cut or a non-normal term;*

2. Π *contains no free term variable;*

3. Π *has no open assumptions;*

Then Π *is either atomic or ends with a* $\exists\mathrm{I}$ *rule instance. In particular, if* Π *is closed, normal and* A *is simply existential then* Π *ends with an introduction.*

Proof. The hypotheses rule out most of the cases considered by Lemma 4. The only possible cases are:

1. Π ends with an introduction rule instance,

2. Π is atomic.

Since A is simple it is either an atomic or existentially quantified formula. If A is atomic, Π cannot end with an introduction rule instance and thus Π must be atomic. Otherwise, if A is existentially quantified, Π cannot end with an atomic rule instance and thus Π must end with an introduction rule instance which can only be a $\exists\mathrm{I}$ rule instance. □

By Theorem 3, the reduction of a derivation halts after a finite number of steps and produces a derivation without head-cuts. Then, Theorem 4 shows that our proof reduction can extract a witness from the derivation of a closed formula $\exists x.\ P$, which can be found in the premise of the $\exists\mathrm{I}$ rule instance at the end of the normalized derivation, by the definition of the $\exists\mathrm{I}$ rule.

Figure 4.2: The Wit−red reduction possible outcomes.

Wit−red

$$\forall\mathrm{E}\,\frac{[\forall x.\ P]^{\alpha}}{P[x := t_1]}\,\beta_1 \cdots \forall\mathrm{E}\,\frac{[\forall x.\ P]^{\alpha}}{P[x := t_n]}\,\beta_n \cdots [\forall x.\ P]^{\alpha}$$

$$\mathsf{EM}_1\,\frac{\dfrac{\Pi_1}{A} \qquad \qquad \dfrac{\begin{array}{c}[P[x := y]]^{\alpha}\\ \Pi_2\end{array}}{A}}{A}\,\alpha$$

An derivation ending with an EM_1 rule instance reduces to:

$$\mathcal{A}\mathrm{I}\,\frac{}{P[x := t_1]} \cdots \mathcal{A}\mathrm{I}\,\frac{}{P[x := t_n]} \cdots [\forall x.\ P]^{\alpha}$$

$$\mathsf{EM}_1\,\frac{\dfrac{\Pi_1}{A} \qquad \qquad \dfrac{\begin{array}{c}[P[x := y]]^{\alpha}\\ \Pi_2\end{array}}{A}}{A}\,\alpha$$

when all $P[x := t_i]$ hold and some occurrences of the universal assumption remain.

$$\mathcal{A}\mathrm{I}\,\frac{}{P[x := t_1]} \cdots \mathcal{A}\mathrm{I}\,\frac{}{P[x := t_n]}$$

$$\frac{\Pi_1}{A}$$

when all $P[x := t_i]$ hold and no occurrence of the universal assumption remains.

$$\dfrac{\begin{array}{c}\to\mathrm{I}\,\dfrac{\mathcal{A}\mathrm{E}\,\dfrac{[P[x := t_i]]^{\beta'}}{\bot}}{\neg P[x := t_i]}\,\beta'\\ \Pi_2\end{array}}{A}$$

when some $P[x := t_i]$ does not hold.

Figure 4.3: The permutative reductions of the EM_1 rule with the $\wedge$E, $\vee$E and $\to$ E rules.

$\mathsf{EM}_1/\wedge$−perm

$$\wedge\mathrm{E}\,\dfrac{\mathsf{EM}_1\,\dfrac{\dfrac{\begin{array}{c}[\forall x.\, P]^{\alpha}\\ \Pi_1\end{array}}{A_1 \wedge A_2} \qquad \dfrac{\begin{array}{c}[\neg P[x := y]]^{\alpha}\\ \Pi_2\end{array}}{A_1 \wedge A_2}}{A_1 \wedge A_2}\,\alpha}{A_i} \quad \to \quad \mathsf{EM}_1\,\dfrac{\wedge\mathrm{E}\,\dfrac{\dfrac{\begin{array}{c}[\forall x.\, P]^{\alpha}\\ \Pi_1\end{array}}{A_1 \wedge A_2}}{A_i} \qquad \wedge\mathrm{E}\,\dfrac{\dfrac{\begin{array}{c}[\neg P[x := y]]^{\alpha}\\ \Pi_2\end{array}}{A_1 \wedge A_2}}{A_i}}{A_i}\,\alpha$$

$\mathsf{EM}_1/\vee$−perm

$$\vee\mathrm{E}\,\dfrac{\mathsf{EM}_1\,\dfrac{\dfrac{\begin{array}{c}[\forall x.\, P]^{\beta}\\ \Sigma_1\end{array}}{A_1 \vee A_2} \qquad \dfrac{\begin{array}{c}[\neg P[x := y]]^{\beta}\\ \Sigma_2\end{array}}{A_1 \vee A_2}}{A_1 \vee A_2}\,\beta \qquad \dfrac{\begin{array}{c}[A_1]^{\alpha}\\ \Pi_1\end{array}}{B} \qquad \dfrac{\begin{array}{c}[A_2]^{\alpha}\\ \Pi_2\end{array}}{B}}{B}\,\alpha$$

$$\downarrow$$

$$\mathsf{EM}_1\,\dfrac{\vee\mathrm{E}\,\dfrac{\dfrac{\begin{array}{c}[\forall x.\, P]^{\beta}\\ \Sigma_1\end{array}}{A_1 \vee A_2} \qquad \dfrac{\begin{array}{c}[A_1]^{\alpha}\\ \Pi_1\end{array}}{B} \qquad \dfrac{\begin{array}{c}[A_2]^{\alpha}\\ \Pi_2\end{array}}{B}}{B}\,\alpha \qquad \vee\mathrm{E}\,\dfrac{\dfrac{\begin{array}{c}[\neg P[x := y]]^{\beta}\\ \Sigma_2\end{array}}{A_1 \vee A_2} \qquad \dfrac{\begin{array}{c}[A_1]^{\alpha}\\ \Pi_1\end{array}}{B} \qquad \dfrac{\begin{array}{c}[A_2]^{\alpha}\\ \Pi_2\end{array}}{B}}{B}\,\alpha}{B}\,\beta$$

$\mathsf{EM}_1/\to$−perm

$$\wedge\mathrm{E}\,\dfrac{\mathsf{EM}_1\,\dfrac{\dfrac{\begin{array}{c}[\forall x.\, P]^{\alpha}\\ \Pi_1\end{array}}{A_1 \to A_2} \qquad \dfrac{\begin{array}{c}[\neg P[x := y]]^{\alpha}\\ \Pi_2\end{array}}{A_1 \to A_2}}{A_1 \to A_2}\,\alpha \qquad \dfrac{\Sigma}{A_1}}{A_2}$$

$$\downarrow$$

$$\mathsf{EM}_1\,\dfrac{\to\mathrm{E}\,\dfrac{\dfrac{\begin{array}{c}[\forall x.\, P]^{\alpha}\\ \Pi_1\end{array}}{A_1 \to A_2} \qquad \dfrac{\Sigma}{A_1}}{A_2} \qquad \to\mathrm{E}\,\dfrac{\dfrac{\begin{array}{c}[\neg P[x := y]]^{\alpha}\\ \Pi_2\end{array}}{A_1 \to A_2} \qquad \dfrac{\Sigma}{A_1}}{A_2}}{A_2}\,\alpha$$

Figure 4.4: The permutative reductions of the EM_1 rule with the $\forall$E and $\exists$E rules.

$\mathsf{EM}_1/\forall$−perm

$$
\forall\mathrm{E}\,\dfrac{\mathsf{EM}_1\,\dfrac{\begin{array}{c}[\forall x.\,P]^\alpha\\ \Pi_1\\ \hline \forall x.\,A\end{array}\qquad \begin{array}{c}[\neg P[x:=y]]^\alpha\\ \Pi_2\\ \hline \forall x.\,A\end{array}}{\forall x.\,A}\,\alpha}{A[x:=t]}
\quad\rightsquigarrow\quad
\mathsf{EM}_1\,\dfrac{\forall\mathrm{E}\,\dfrac{\begin{array}{c}[\forall x.\,P]^\alpha\\ \Pi_1\\ \hline \forall x.\,A\end{array}}{A[x:=t]}\qquad \forall\mathrm{E}\,\dfrac{\begin{array}{c}[\neg P[x:=y]]^\alpha\\ \Pi_2\\ \hline \forall x.\,A\end{array}}{A[x:=t]}}{A[x:=t]}\,\alpha
$$

$\mathsf{EM}_1/\forall$−perm

$$
\exists\mathrm{E}\,\dfrac{[\Sigma]\,\mathsf{EM}_1\,\dfrac{\begin{array}{c}[\forall x.\,P]^\alpha\\ \Pi_1\\ \hline \exists x.\,A\end{array}\qquad \begin{array}{c}[\neg P[x:=y]]^\alpha\\ \Pi_2\\ \hline \exists x.\,A\end{array}}{\exists x.\,A}\,\alpha \qquad \begin{array}{c}[A[x:=y]]\\ B\end{array}}{B}
$$

$$\downarrow$$

$$
\mathsf{EM}_1\,\dfrac{[\Sigma]\,\exists\mathrm{E}\,\dfrac{\begin{array}{c}[\forall x.\,P]^\alpha\\ \Pi_1\\ \hline \exists x.\,A\end{array}\qquad \begin{array}{c}[A[x:=y]]\\ B\end{array}}{B}\qquad \exists\mathrm{E}\,\dfrac{\begin{array}{c}[\neg P[x:=y]]^\alpha\\ \Pi_2\\ \hline \exists x.\,A\end{array}\qquad \begin{array}{c}[A[x:=y]]^\beta\\ \Sigma\\ \hline B\end{array}}{B}\,\beta}{B}\,\alpha
$$

Chapter 5

Interpreting a Geometric Example with Interactive Realizability

In this chapter we show how to extract a monotonic learning algorithm from a classical proof of a geometric statement by interpreting the proof by means of interactive realizability.

The statement is about the existence of a convex angle including a finite collections of points in the real plane and it is related to the existence of a convex hull. We define real numbers as Cauchy sequences of rational numbers, therefore equality and ordering are not decidable. While the proof looks superficially constructive, it employs classical reasoning to handle undecidable comparisons between real numbers, making the underlying algorithm non-effective.

The interactive realizability interpretation transform the non-effective linear algorithm described by the proof into an effective one that uses backtracking to learn from its mistakes. The effective algorithm exhibit a "smart" behavior, performing comparisons only up to the precision required to prove the final statement. This behavior is not explicitly planned but arises from the interactive interpretation of comparisons between Cauchy sequences.

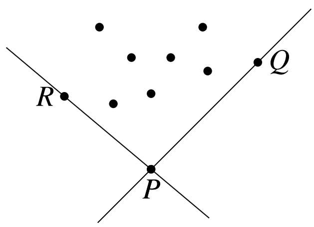
5.1 Introduction

We study the computational content of the proof of the following geometric statement.

Theorem 5 (Convex Angle)**.** *We have a finite set of at least three points in the real plane*

$\mathbb{R}^2$ such that no three points are on the same line. Then there exist distinct points P, Q and R such that:

- *all other points S are inside $\widehat{QPR}$,*
- *the angle $\widehat{QPR}$ is convex, that is, less than π.*



We choose this particular statement because we have a proof of it that looks algorithmic and can be easily visualized. The Convex Angle Theorem be thought of as weakened version of the existence of the convex hull of a finite set of points.

As we said proof we choose as example looks constructive, using only decidability of ordering over real numbers. However, it is well known that there is no effective ordering on the real numbers. In our encoding of the real numbers, totality of the ordering on the recursive reals is equivalent to EM_1. Since the proof needs the ordering to be total, it needs EM_1. Due to the low logical complexity of excluded middle which is used, the proof may be interpreted with a simple case of interactive realizability.

We show how interactive realizability can be applied and what it can tell us about the computational content of the proof. What we get is an algorithm that, instead of comparing real numbers, makes an arbitrary guess about which one is smaller. If later it becomes apparent that the guess is wrong the algorithm retracts the choice it made since it can now make an informed decision about that particular comparison. Then the algorithm performs comparisons only when needed and only up to the required precision.

Thus we see how a simple classical proof which performs comparisons between real numbers is interpreted as a learning algorithm which uses "educated guesses" in order to avoid non effective operations. This non-trivial behavior is not explicit in the classical proof, but follows from the definition of ordering on Cauchy sequences by means of the interactive realizability interpretation.

In this chapter, our main goal is to showcase interactive realizability and the backtracking algorithms it produces through a non-trivial example. For this reason, we chose to present interactive realizability as a proof interpretation technique rather than as a realizability semantics, in order to concentrate on the example and its computational interpretation without being bogged down in technical details.

Note that interactive realizability is by no means the only approach to extract a computational interpretation from our proof. It should also be noted that while our proof is classical, it can be seen that our statement admits an intuitionistic proof by the conservativity results in [8].

5.2 Real Numbers

In this section we present our treatment of real numbers in Heyting Arithmetic.

There are many ways of encoding integer and rational numbers in HA and defining primitive recursive operations and predicates on them. In the following we assume that we have any such encoding and that we have decidable equality $=_{\mathbb{Q}}$ and ordering $<_{\mathbb{Q}}, \leq_{\mathbb{Q}}$ and effective operations $+_{\mathbb{Q}}\ \cdot_{\mathbb{Q}}$. We use the variables q and p for rationals.

5.2.1 Cauchy Sequences

There are many equivalent ways of defining the real numbers from the rational numbers. The most known are the definition of the reals as equivalence classes of Cauchy sequences and as Dedekind cuts. We follow the first approach.

A sequence of rationals $r : \mathbb{N} \to \mathbb{Q}$ is a *Cauchy sequence* if the following holds:

$$\forall k.\ \exists k_0.\ \forall k_1, k_2.\ |r(k_0 + k_2) - r(k_0 + k_1)| < \frac{1}{2^k}. \tag{5.1}$$

While this sequence approximates a real number, it can do so very slowly. By means of classical reasoning, we can show that, from any Cauchy sequence, we can extract a fast-converging monotone sub-sequence. For this reason, instead of general Cauchy sequences, we can consider sequences of nested intervals with rational extremes whose length decreases exponentially. An interval is determined by its extremes, so we represent a sequence of intervals as a couple of sequences of rationals r^-, r^+, representing the lower and higher extremes

of the intervals respectively. Then we require that r^- is increasing and r^+ is decreasing (since the intervals are nested), that $r^-(k)$ is lesser than or equal to $r^+(k)$ (since they are the lower and higher extremes of a same interval) and their difference is smaller than 2^{-k}. More precisely we say that r^- and r^+ represent a real number when they satisfy the following condition, written as a Π^0_1 formula:

$$\begin{aligned} \forall k.\ (r^-(k) \leq_{\mathbb{Q}} r^+(k)) \wedge (r^-(k) \leq_{\mathbb{Q}} r^-(k+1)) \wedge \\ \wedge\ (r^+(k) \geq_{\mathbb{Q}} r^+(k+1)) \wedge (r^+(k) -_{\mathbb{Q}} r^-(k) \leq_{\mathbb{Q}} 2^{(-k)}). \end{aligned} \tag{5.2}$$

While the choice of the specific definition of real number is somewhat arbitrary, it is significant because it affects the logical properties (in particular the degree of undecidability) of the ordering on the reals.

5.2.2 Order Predicate

Now we can define an "order predicate" $\mathrm{OP}(r, s, k)$, which can be thought of as a family of strict partial orders on the real numbers indexed by natural number k. More precisely, it is a formula that determines when the sequence of nested intervals r is strictly lesser than s, at precision k. This happens when, at k, the higher extreme of an interval is strictly greater than the lower extreme of the other. Then, from that point forward, the intervals will be forever disjoint, since we they are nested sequences. This allows us to write the order predicate as the formula:

$$\mathrm{OP}(r, s, k) \equiv r^+(k) <_{\mathbb{Q}} s^-(k), \tag{5.3}$$

which is decidable in r and s. Note that the definition of OP depends on that of real number. If we had used the classical definition of Cauchy sequence the order predicate would be the following Π^0_1 formula:

$$\mathrm{OP}'(r, s, k) \equiv \forall l.\ l \geq k \rightarrow r(l) <_{\mathbb{Q}} r(l). \tag{5.4}$$

This is very significant for our purposes: the order predicate in (5.3) is decidable in r and s (since the order on the rationals is), while in (5.4) it is only *negatively decidable*. This means that we have an effective method to decide (5.4) when it is false, but not when it is true.

We need OP to satisfy some properties, written as rules in Figure 5.1. The OP-mon

Figure 5.1: Rules for OP.

$$\text{OP-mon}\ \frac{\mathrm{OP}(r,s,k)}{\mathrm{OP}(r,s,k+1)} \qquad \text{OP-irrefl}\ \frac{\mathrm{OP}(r,r,k)}{\bot}$$

$$\text{OP-asym}\ \frac{\mathrm{OP}(r,s,k) \qquad \mathrm{OP}(s,r,l)}{\bot} \qquad \text{OP-trans}\ \frac{\mathrm{OP}(r,s,k) \qquad \mathrm{OP}(s,t,l)}{\mathrm{OP}(r,t,\max(k,l))}$$

rule expresses a monotonicity property: when an comparison at a given precision can distinguish two approximations, then comparisons at greater precision should too. The other rules correspond to the standard axioms for a strict partial order: irreflexivity, asymmetry and transitivity.

We verify that our definition of OP satisfies these properties.

Lemma 5. *The order predicate OP defined by* (5.3) *satisfies the properties given in Figure 5.1.*

Proof. We show that the properties follow directly from the definition of OP as (5.3) and from our representation of real number as sequences of nested intervals (5.2).

Monotonicity We want to prove that

$$\mathrm{OP}(r,s,k+1) \equiv r^+(k+1) <_{\mathbb{Q}} s^-(k+1),$$

assuming that:

$$\mathrm{OP}(r,s,k) \equiv r^+(k) <_{\mathbb{Q}} s^-(k).$$

This follows by applying the transitive property of the order on the rationals to the following chain of inequalities:

$$\begin{aligned} r^+(k+1) &\leq_{\mathbb{Q}} r^+(k) && \text{since } r^+ \text{ is weakly decreasing,} \\ &<_{\mathbb{Q}} s^-(k) && \text{by assumption,} \\ &\leq_{\mathbb{Q}} s^-(k+1) && \text{since } s^+ \text{ is weakly increasing.} \end{aligned}$$

Reflexivity We have to prove that

$$\mathrm{OP}(r, r, k) \equiv r^+(k) <_{\mathbb{Q}} r^-(k),$$

yields a contradiction. This is a consequence of the fact that $r^-(k)$ and $r^+(k)$ are respectively the lower and higher extremes of the same interval.

Asymmetry The OP-asym is actually derivable by monotonicity and transitivity:

$$\text{OP-irrefl}\ \dfrac{\text{OP-trans}\ \dfrac{\mathrm{OP}(r, s, k) \qquad \mathrm{OP}(s, r, l)}{\mathrm{OP}(r, r, k)}}{\bot}$$

Transitivity We have to prove that

$$\mathrm{OP}(r, t, k) \equiv r^+(k) <_{\mathbb{Q}} t^-(\max(k, l)),$$

follows from the assumptions:

$$\mathrm{OP}(r, s, k) \equiv r^+(k) <_{\mathbb{Q}} s^-(k),$$
$$\mathrm{OP}(s, t, l) \equiv s^+(l) <_{\mathbb{Q}} t^-(l).$$

We have two cases depending on whether $\max(k, l)$ is k or l. Since the two cases are very similar, we only show the proof of the first. Thus we assume that $\max(k, l) = k$, which means that $k \geq k$.

Again this follows applying the transitive property of the order on the rationals to the following chain of inequalities:

$$\begin{aligned}
r^+(k) &<_{\mathbb{Q}} s^-(k) && \text{by the first assumption,}\\
&\leq_{\mathbb{Q}} s^+(k) && \text{since } [s^-(k), s^-(k)] \text{ is an interval,}\\
&\leq_{\mathbb{Q}} s^+(l) && \text{since } s^+ \text{ is weakly decreasing and } k \geq l,\\
&<_{\mathbb{Q}} t^-(l) && \text{by the second assumption,}\\
&\leq_{\mathbb{Q}} t^-(k) && \text{since } t^- \text{ is weakly increasing and } k \geq l.
\end{aligned}$$

Thus, we have:

$$r^+(k) <_{\mathbb{Q}} t^-(\max(k, l)). \qquad \square$$

5.2.3 Order and Equality on the Real Numbers

We can now defined order and equality on the reals. It is noteworthy that, while we define order and equality in terms of OP, we never use the definition of OP itself in proving their properties. We only need the properties of OP we proved in Lemma 5, thus we could proceed in the same way even if we had defined OP differently, as long as Lemma 5 holds.

They are defined as follows:

$$\begin{aligned}
r <_{\mathbb{R}} s &\equiv \exists k.\ \mathrm{OP}(r, s, k),\\
r \leq_{\mathbb{R}} s &\equiv \forall k.\ \neg\mathrm{OP}(s, r, k),\\
r \neq_{\mathbb{R}} s &\equiv \exists k.\ \mathrm{OP}(r, s, k) \vee \mathrm{OP}(s, r, k),\\
r =_{\mathbb{R}} s &\equiv \forall k.\ \neg\mathrm{OP}(r, s, k) \wedge \neg\mathrm{OP}(s, r, k).
\end{aligned}$$

Note that $<_{\mathbb{R}}$ and $\neq_{\mathbb{R}}$ are Σ^0_1 formulas and $\leq_{\mathbb{R}}$ and $=_{\mathbb{R}}$ are Π^0_1 formulas. Moreover $\leq_{\mathbb{R}}$ and $=_{\mathbb{R}}$ are the dual formulas of $<_{\mathbb{R}}$ and $\neq_{\mathbb{R}}$ respectively, as defined in Section 2.1.6.

In order to prove the Least Element Lemma, which is needed in the proof of the Convex Angle Theorem, we need to show some of the properties of the order $\leq_{\mathbb{R}}$.

Lemma 6 (Reflexivity, Semi-Transitivity and Totality of $\leq_{\mathbb{R}}$)**.** *The following properties hold:*

$$r \leq_{\mathbb{R}} r \qquad \text{(reflexivity)}$$

$$r <_{\mathbb{R}} s \wedge s \leq_{\mathbb{R}} t \to r \leq_{\mathbb{R}} t, \qquad \text{(semi-transitivity)}$$

$$r \leq_{\mathbb{R}} s \vee s <_{\mathbb{R}} r. \qquad \text{(totality)}$$

Proof. The first two properties follows from the corresponding properties of OP. The last is a classical tautology.

- In order to prove reflexivity we have to show that:

$$r \leq_{\mathbb{R}} r \equiv \forall k.\ \neg\mathrm{OP}(r, r, k).$$

  This follows by the OP-irrefl rule:

$$\forall\mathrm{I}\,\dfrac{\to\mathrm{I}\,\dfrac{\text{OP-irrefl}\,\dfrac{[\mathrm{OP}(r, r, k)]^{\alpha}}{\bot}}{\neg\mathrm{OP}(r, r, k)}\,\alpha}{\forall k.\ \neg\mathrm{OP}(r, r, k)}$$

- In order to prove this transitive property for mixed $<_{\mathbb{R}}$ and $\leq_{\mathbb{R}}$ we have to show that:

$$r \leq_{\mathbb{R}} t \equiv \forall k.\ \neg \mathrm{OP}(t, r, k),$$

  assuming that:

$$r <_{\mathbb{R}} s \equiv \exists k.\ \mathrm{OP}(r, s, k),$$
$$s \leq_{\mathbb{R}} t \equiv \forall k.\ \neg \mathrm{OP}(t, s, k).$$

  This follows by means of the OP-trans rule:

$$\forall\mathrm{I}\,\dfrac{\to\mathrm{I}\,\dfrac{\exists\mathrm{E}\,\dfrac{\exists k.\ \mathrm{OP}(r,s,k) \qquad \to\mathrm{E}\,\dfrac{\forall\mathrm{E}\,\dfrac{\forall k.\ \neg\mathrm{OP}(t,s,k)}{\neg\mathrm{OP}(t,s,\max(k,l))} \qquad \text{OP-trans}\,\dfrac{[\mathrm{OP}(t,r,k)]^1 \qquad [\mathrm{OP}(r,s,\bar{k})]^2}{\mathrm{OP}(t,s,\max(k,l))}}{\bot}}{\bot}\,2}{\neg\mathrm{OP}(t,r,k)}\,1}{\forall k.\ \neg\mathrm{OP}(t,r,k)}$$

- We have to show that:

$$r \leq_{\mathbb{R}} s \vee s <_{\mathbb{R}} r \equiv \forall k.\ \neg \mathrm{OP}(s, r, k) \vee \exists k.\ \mathrm{OP}(r, s, k),$$

  which is an instance of EM_1 when r and s denote recursive real numbers. □

The proof is constructive apart from the last point, where we show that totality is actually an instance of EM_1. Note that only the reflexivity property is stated in the standard way, while transitivity and totality are written in non-standard forms. We chose these forms for two reasons: they are easier to prove and they are the exact form we need in the proof of the Least Element Lemma.

5.2.4 Variables for Real Numbers

Until now we have used r, s and t as metavariables for real numbers in an informal way. However, since we are working in the first-order language of arithmetic, our variables range only on natural numbers and not on functions. For our example we only need to address a finite but arbitrary number of real numbers, that is, we only need a countable quantity of them. Thus we can assume that we have a countable set of function symbols indexed by the

natural numbers. These function symbols represent sequences of rational numbers satisfying some notion of convergence.

In the case we are considering, where real numbers are represented with sequences of nested intervals satisfying the convergence condition (5.2), we proceed as follows. We assume that we have two sequences of indexed function symbols:

$$f_0^+, \ldots, f_n^+, \ldots \quad \text{and} \quad f_0^-, \ldots, f_n^-, \ldots$$

such that, for any index n, f_n^+ and f_n^- satisfy the convergence condition (5.2). Then we can formally define the order predicate as:

$$\mathrm{OP}(i, j, k) \equiv f_i^+(k) <_{\mathbb{Q}} f_j^-(k),$$

where i and j are metavariables for arithmetic terms. Thus, when we write $i < j$, we mean that i is smaller than s as indexes, that is, as natural numbers; on the other hand, when we write $i <_{\mathbb{R}} j$, we mean that the real number indexed by i is smaller than the one indexed by j.

However this notation, while formally correct, is hard to read: $i \leq j$ and $i \leq_{\mathbb{R}} j$ look confusingly similar while having unrelated meaning. In order to avoid confusion and hurting the eyes of mathematicians, we sugar coat our syntax. We write r_i instead of i when thinking of i as a real number. For instance we write $r_i \leq_{\mathbb{R}} r_j$ instead of $i \leq_{\mathbb{R}} j$. The last one is a much more intuitive than the unsugared version.

5.2.5 The Least Element Lemma

Now we can reason about finite sets of real numbers as sets of indexes. In the next lemma, we shall work with the sets of real numbers indexed by initial segments of the natural numbers. We show the existence of a least element in each of these sets. The least element is actually a minimum, that is, the unique least element of the set. However, in order to prove the Convex Angle Theorem we do not need to show its uniqueness, just its existence.

Lemma 7 (Least Element)**.** *For any n, the real numbers $r_0, \ldots, r_n$ have a least element with*

respect to $\leq_{\mathbb{R}}$. More precisely: [1]

$$\forall n.\ \exists i \leq n.\ \forall j \leq n.\ r_i \leq_{\mathbb{R}} r_j.$$

Proof. We proceed by induction on n.

Zero case In the base case $n = 0$ and we have to prove that:

$$\exists i \leq 0.\ \forall j \leq 0.\ r_i \leq_{\mathbb{R}} r_j.$$

Both i and j can only be 0; thus we just have to check the condition $r_0 \leq_{\mathbb{R}} r_0$, which holds by reflexivity of $\leq_{\mathbb{R}}$.

Successor case In the inductive case we have to prove that:

$$\exists i \leq n+1.\ \forall j \leq n+1.\ r_i \leq_{\mathbb{R}} r_j,$$

from the inductive hypothesis:

$$\exists i \leq n.\ \forall j \leq n.\ r_i \leq_{\mathbb{R}} r_j.$$

By the inductive hypothesis, let $\bar{i} \leq n$ be the index of the least element in $r_0, \ldots, r_n$. By totality of $\leq_{\mathbb{R}}$ we have two cases.

$r_{\bar{i}} \leq_{\mathbb{R}} r_{n+1}$ Then $\bar{i}$ is the index of a least element in $r_0, \ldots, r_{n+1}$, since $r_{\bar{i}} \leq_{\mathbb{R}} r_j$ when $j = n+1$ (since we are considering this case) and when $j \leq n$ by inductive hypothesis.

$r_{n+1} <_{\mathbb{R}} r_{\bar{i}}$ Then $n+1$ is the index of a least element in $r_0, \ldots, r_{n+1}$, since $r_{n+1} \leq_{\mathbb{R}} r_j$ when $j = n+1$ by reflexivity of $\leq_{\mathbb{R}}$ and when $j \leq n$ by transitivity of $<_{\mathbb{R}}$ and $\leq_{\mathbb{R}}$, since:

$$r_{n+1} <_{\mathbb{R}} r_{\bar{i}} \leq_{\mathbb{R}} r_j$$

by inductive hypothesis. □

[1] We use the standard compact notation for bounded quantifications:

$$\forall j \leq n.\ A \text{ stands for } \forall j.\ j \leq n \to A,$$
$$\exists j \leq n.\ A \text{ stands for } \exists j.\ j \leq n \wedge A.$$

The proof looks constructive: its computational interpretation is the usual algorithm that finds the least element in a vector, by a simple recursion or by looping on its elements. We can write it as a recursive function "`rmin`" in Haskell:

Listing 5.1: The Least Element Program

```
rmin 0   = 0
rmin n   = if rle (rmin (n-1)) n
  then rmin (n-1)
  else n
```

where "`rle`" is a boolean function that stands for $\leq_{\mathbb{R}}$, that is, it compares the reals indexed by its arguments. The problem is that this is not a good program, because we are unable to write "`rle`" as a terminating program. The closest approximation would be the following unfounded recursion:

Listing 5.2: The Lesser or Equal Program

```
rle i j        = rle_urec 0 i j
rle_urec k i j = if op j i k
  then False
  else rle_urec (k+1) i j
```

where "`op`" is a total boolean function that stands for the order predicate OP. We can assume that "`op`" terminates for any input since OP is decidable. The problem is that $\leq_{\mathbb{R}}$ is total only classically. More precisely, totality is an instance of EM_1 because $\leq_{\mathbb{R}}$ is a Π^0_1 formula and thus negatively decidable. This can bee seen concretely in the program for "`rle`": "`rle i j`" only halts (returning "**`False`**") if "`op j i k`" is true for some k, that is, if and only if $r_i \leq r_j$ is false. On the other hand, when $r_i \leq r_j$ is true there is no such k and the evaluation of "`rle i j`" will never halt. Note that "**`True`**" does not even occur in the program, so its is clear that "`rle i j`" never returns "**`True`**". This is the general behavior of an algorithm that computes a negatively decidable predicate: when the predicate is false it halt with the correct answer and when the predicate is true it does not halt.

For positively decidable predicates we have the dual behavior. For instance, in the case of $<_{\mathbb{R}}$ which is defined by a Σ^0_1 formula and thus positively decidable, the decision procedure can be written as:

Listing 5.3: The Lesser Than Program

```
rlt i j        = rlt_urec 0 i j
rlt_urec k i j = if op i j k
  then True
  else rlt_urec (k+1) i j
```

The program is very similar to the previous one, the only noteworthy changes are the order of the argument given to "`op`" and the fact that the only possible return value is "`True`" instead of "`False`". It only halts (returning "`True`") if "`op i j k`" is true for some k, that is, when $r_i \leq r_j$ is true.

Remark 3. *Note how Program 5.1 is much shorter than the proof of the Least Element Lemma. This difference would be even bigger if the proof was written in a completely formal language, for instance in a proof assistant. The reason for this discrepancy is that the proof contains both the algorithm written in Program 5.1 and the evidence for the correctness of the algorithm. This last part is missing from Program 5.1, thus explaining the difference in length.*

5.3 The Interactive Interpretation of the Least Element Lemma

We have seen why the naive way of extracting a program from proofs fails in the case of the Least Element Lemma. Now we give the interactive interpretation of the Least Element Lemma. Since we are working in $\mathsf{HA} + \mathsf{EM}_1$, any proof can be thought of as a constructive proof with open assumptions, that are the instances of EM_1 that are used in the proof. The interactive realizability interpretation follows the standard BHK interpretation for the constructive parts, so we will concentrate on the interpretation of the EM_1 instances.

The only instances of EM_1 in the proof are those used to deduce the totality property:

$$r_i \leq_{\mathbb{R}} r_j \vee r_j <_{\mathbb{R}} r_i. \tag{5.5}$$

The left disjunct, which we call the *universal disjunct*, is Π^0_1 and negatively decidable, while the right one, the *existential disjunct* is Σ^0_1 and positively decidable. Moreover universal disjunct and negation of the existential disjunct are classically equivalent. We say that a formula is *concrete* when it is closed and all its arithmetical terms are normal.

In order to motivate the interactive realizability interpretation we show why a naive attempt to give a computational behavior an EM_1 instance fails. This can be seen concretely, by recalling the semi-effective procedures Programs 5.2 and 5.3 that decide the disjuncts of the instances of EM_1 representing the totality, but the general argument is the same. The universal disjunct is negatively decidable, that is, its deciding program halts if and only if it is false; the existential disjunct is positively decidable, that is, its deciding program halts if and only if it is true. What happens if we run these two decision programs in parallel? Can one give the answer whenever the other fails? In recursion theory, this method is used for instance to prove that if the complement of a recursively enumerable set is recursively enumerable then the set is recursive. Unfortunately this does not work in our case. We have two scenarios:

- If the existential disjunct is true, its decision procedure halts and returns true. Since in this case the universal disjunct is false, its decision procedure also halts and returns false.

- If the universal disjunct is true, its decision procedure does not halt. Since in this case the existential disjunct is false, its decision procedure does not halt either.

The problem lies in the fact that the disjuncts are dual and their decision procedures describe basically the same algorithm with minor variations. In particular they halt or fail to halt on the same inputs. This is evident when considering the programs given in Programs 5.2 and 5.3.

Interactive realizability proposes a way to side-step the problem evidenced above. This is possible since it is not true that the computational interpretation of a proof using instances of EM_1 necessarily needs to decide these instances. Consider the case of totality of the order on the real numbers. The universal disjunct is:

$$r_i \leq_{\mathbb{R}} r_j \equiv \forall k.\ \neg\mathrm{OP}(r_j, r_i, k).$$

Being an universally quantified statement, it proves infinite instances $\neg\mathrm{OP}(r_j, r_i, k)$, one for each natural number k. A proof that uses totality may need all this infinite information or (for example, when proving a simply existential statement) may only need a finite quantity of these instances. In the second case, we can avoid the problem of effectively deciding the

EM_1 instance. We only need to decide those instances that are actually used in the proof. This is possible, since each instance is decidable (being a quantifier free formula) and we assumed there is a finite quantity of them. Interactive realizability takes advantage of this fact and gives a procedure to determine which instances of the universal disjunct are needed and to iteratively decide them.

The interactive interpretation is a "relaxation" of the BHK interpretation. In the BHK interpretation the decision of a disjunction effectively selects a true disjunct, in the interactive case instead of a decision we have a sort of "educated guess". Therefore, while EM_1 cannot be realized by the BHK interpretation since there is no effective procedure to decide it, the interactive interpretation can because it yields a weaker semantics, which produces a sure result only when the goal is simply existential.

Interactive realizability revolves around the concept of *knowledge state*. A knowledge state, or simply state, is a finite object that stores information about the EM_1 instances we use in the proof. The purpose of this information is help us decide the EM_1 instances, that is, help us in choosing which disjunct holds. Moreover, whenever the state chooses the existential disjunct, it should also produce a witness, like in the BHK interpretation.

We can represent a state as a finite partial function[2] that maps a concrete instance of EM_1 into a witness of its existential disjunct. Such a function decides or guesses a concrete instance A of EM_1: if it is undefined on A, then we choose the universal disjunct; if it is defined we chose the existential disjunct with the returned witness. We are only interested to the instances appearing in the proof, namely, those of the form (5.5) when i, j are numerals. Thus an instance is determined by two natural numbers; since witnesses are natural numbers too, a state can be concretely defined as a finite partial function from $\mathbb{N} \times \mathbb{N}$ to $\mathbb{N}$.

For instance, consider the case of the EM_1 instances used in the proof of the Least Element Lemma. When we have to decide (5.5), we check the state on the pair (i, j). At first, let us assume that the state is undefined on (i, j). This means we have no knowledge about the universal disjunct $r_i \leq_{\mathbb{R}} r_j$. Since we cannot effectively check that the universal disjunct holds, we make an educated guess and assume that $r_i \leq_{\mathbb{R}} r_j$ is true. Clearly this assumption could very well be wrong, which may or may not become apparent later in the proof. Keeping track of this assumption, we carry on with the proof. Every time we use this assumption

[2]By finite partial function, we mean a partial function whose domain (the set of elements where it is defined) is finite.

to prove a decidable instance of its we check if the instance holds. More concretely, if later in the proof we use the assumption $r_i \leq_{\mathbb{R}} r_j$ to deduce that $\neg\mathrm{OP}(j, i, k)$ for some k, we check that $\neg\mathrm{OP}(j, i, k)$ holds. If this is the case, we carry on with the proof: $r_i \leq_{\mathbb{R}} r_j$ could still be false, but at least the particular instance we are using is true. If this is not the case, we have found a counterexample to the assumption $r_i \leq_{\mathbb{R}} r_j$: being negatively decidable, the counterexample is enough to effectively decide that it is false. Therefore we stop following the proof because we have chosen the wrong disjunct in the EM_1 instance (5.5).

Moreover, a counterexample to $r_i \leq_{\mathbb{R}} r_j$ is a natural number k such that $\mathrm{OP}(j, i, k)$. Therefore k is a witness for the existential disjunct $r_j <_{\mathbb{R}} r_i$. We can use this new knowledge to add (i, j) to the domain of the state with value k. Remember that we assumed the state to be undefined on (i, j), which is why we assumed the universal disjunct to be true in the first place.

At this point, we forget what we did after guessing (wrongly) that the universal disjunct was true and start again, More precisely, we need to backtrack to a computation state *before* we decided the EM_1 instance in question and repeat our decision with the extended state. Since the extended state is defined on (i, j) and yields k, this time we decide the EM_1 instance differently: we choose the existential disjunct $r_j <_{\mathbb{R}} r_i$ with k as witness. Now we are sure that our choice is the correct one and not a guess, since we have effectively decided that the existential disjunct holds (we can since it is positively decidable).

The exact point we need to backtrack to is not relevant, as long as it is before the decision of the EM_1 instance. A simple choice would be the very beginning, in which case we do not need to keep track of where we decided the EM_1 instance. A more efficient choice is right before the decision point, so that we do not need to repeat the computations before it, which do not change.

In order for the interactive interpretation to produce correct results, we need to assume that the state is sound, that is, when it is defined, the witness it yields is actually a witness. More formally, a state s is sound if, for any pair (i, j), we have that $\mathrm{OP}(j, i, s(i, j))$ holds. This assumption is not problematic: the empty state, namely the state that is always undefined, satisfies it vacuously. Moreover, note that in the interactive interpretation we outlined above, we only extend a state with an actual witness. In other words, the extension preserves the soundness property.

To summarize, the general procedure is the following:

1. we start from any sound state (usually the empty state),
2. we follow the proof choosing any EM_1 instance according to the state,
3. if we discover that we wrongly assumed the universal disjunct of an EM_1 instance:
   (a) we extend the state with the counterexample we found,
   (b) we backtrack to a point before the EM_1 instance we guessed wrong,
   (c) we proceed as in step 2,
4. if we never discover that we wrongly assumed an universal disjunct we carry on until the end of the proof and we are done.

Interactive realizability can be thought as a "smart", albeit "partial", decision algorithm for negatively decidable statements. This can be seen comparing it with the naive algorithm given in Program 5.2. It is partial because a real decision is impossible, so it only considers a finite number of instances, unlike the unbounded recursion employed by Program 5.2. It is smart because it does not perform a blind search, trying in order all the natural numbers. Instead it uses the proof itself to find the counterexamples. There is a reasonable expectation that the ideas underlying the proof provide a more focused way of selecting counterexamples than a blind search (this of course depends on the proof itself).

Until now we considered a single instance of the EM_1 axiom, but little changes if there is more than one. We will return to this point later. In the proof of the Least Element Lemma, one instance of EM_1 is used for each inductive step in the proof. When we interpret the proof with the empty state, for each of these instances we assume that the universal disjunct holds. Therefore the proof is interpreted as follows. In the base step we choose r_0. In the first inductive step, we have to decide the EM_1 instance:

$$r_0 \leq_{\mathbb{R}} r_1 \vee r_1 <_{\mathbb{R}} r_0.$$

Since the state is empty, we assume that $r_0 \leq_{\mathbb{R}} r_1$. Thus we keep r_0 as the least element of r_0, r_1. In the second inductive step, we have to decide the EM_1 instance:

$$r_0 \leq_{\mathbb{R}} r_2 \vee r_2 <_{\mathbb{R}} r_0.$$

Since the state is empty, we again assume that $r_0 \leq_{\mathbb{R}} r_2$. Thus we keep r_0 as the least element of r_0, r_1, r_2. At the end of the proof, we have assumed the following universal disjuncts:

$$r_0 \leq_{\mathbb{R}} r_1, r_0 \leq_{\mathbb{R}} r_2, \ldots, r_0 \leq_{\mathbb{R}} r_n. \tag{5.6}$$

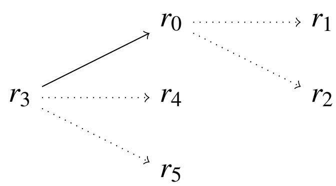
Under these assumptions, we have found that the least element is r_0. Rather disappointing, isn't it?

The reason for this is that the universal disjuncts $r_i \leq_{\mathbb{R}} r_j$ are never instanced, so we have neither opportunity or reason to falsify one of them. However this may change if the Least Element Lemma is used inside a bigger proof. This will happen later in the proof of the Convex Angle Theorem. In this case the outer proof might instance these assumptions and discover them wrong, in which case we have to backtrack to the proof of the Least Element Lemma.

Let us see how the Least Element Lemma behaves when its conclusion is used to deduce decidable instances. Assume that $n = 5$. If the state is empty, then the Least Element Lemma tells us that r_0 is a least element. This means that $r_0 \leq_{\mathbb{R}} r_i$ for any i. Imagine that we use the Least Element Lemma in a bigger proof to prove that $r_0 \leq_{\mathbb{R}} r_3$. This is one of the EM_1 instances we assumed in (5.6). Moreover, imagine that, using this assumption, we discover that $r_0 \leq_{\mathbb{R}} r_3$ does not hold at precision 33. Then we have to extend the domain of the state to $(0, 3)$ with value 33. At this point we backtrack, say at the beginning of the proof of the Least Element Lemma.

We again start from r_0 and proceed like before. The first and second inductive steps again select r_0 as the least element, assuming that $r_0 \leq_{\mathbb{R}} r_1$ and $r_0 \leq_{\mathbb{R}} r_2$. Things change at the third inductive step when we have to decide $r_0 \leq_{\mathbb{R}} r_3 \vee r_3 <_{\mathbb{R}} r_0$. Since know the state has a relevant witness, this time we choose the existential disjunct with witness 33, thus selecting r_3 as the new least element. In the next inductive steps we again assume the universal disjuncts $r_3 \leq_{\mathbb{R}} r_4$ and $r_3 \leq_{\mathbb{R}} r_5$, since the state has no information on them. Thus our the least element is r_3. A summary of our decisions is represented in Figure 5.2. Imagine that we were to discover a counterexample to $r_3 \leq_{\mathbb{R}} r_2$, say at precision 25. This statement is not one of the universal disjuncts that we assumed. By looking at the proof or at Figure 5.2, we can see that it has been deduced by the semi-transitivity property from $r_3 <_{\mathbb{R}} r_0$ and $r_0 \leq_{\mathbb{R}} r_2$. The first is the existential disjunct for which we found a witness, so we are sure that it holds. Thus the the wrong assumption is $r_0 \leq_{\mathbb{R}} r_2$. By checking the proof of semi-

Figure 5.2: A graph representing the result of the least element computation. Full arrows represent information provided by the state, dotted arrows "guessed" information the state knows nothing about.



transitivity we can see that the counterexample for $r_0 \leq_{\mathbb{R}} r_2$ is $\max(25, 33)$, thus 33 again. We extend the state accordingly and repeat the least element computation, which results in new least element r_2. In Figure 5.3 we summarize the iterations we saw until now and add some more, as an example.

5.3.1 Backtracking, Termination and Complexity

In the iterations listed in Figure 5.3, we compute the following sequence of least element candidates:

$$r_0, r_3, r_2, r_3, r_1, r_4.$$

The fact that r_3 appears two times may cause doubts regarding the termination of the backtracking algorithm. The termination of the backtracking algorithms in interactive realizability has been proven in general, see Theorem 2.15 in [2].

In this particular case we can understand why r_3 is computed two times by taking a closer look at the tree of the possible computations of the least element, which is shown in Figure 5.4. For reasons of space, we only show the tree for $n = 3$, which is enough to see what happens up to the fifth iteration in Figure 5.3. We can see that the first five iterations in Figure 5.3 correspond to the computation paths ending with the first five leaves from the left in Figure 5.4, in order.

Moreover, from the computation tree we can see that we never perform the same computation more than once. Indeed, assume we have just followed a particular computation path. When we backtrack we increment the state adding a witness of one of the EM_1 instances

Figure 5.3: An example of evaluations of the interactive interpretation of the Least Element Lemma with state extensions.

Iter	State	Least element	Used	Deduced from	Discovered
$1st$		$r_0 \cdots r_{1,\ldots,5}$	$r_0 \leq_{\mathbb{R}} r_3$	$r_0 \leq_{\mathbb{R}} r_3$	$r_3 <_{\mathbb{R}} r_0$
$2nd$	$r_3 <_{\mathbb{R}} r_0$	$r_3 - r_0 \cdots r_{1,2}$ $r_3 \cdots r_{4,5}$	$r_3 \leq_{\mathbb{R}} r_2$	$r_3 <_{\mathbb{R}} r_0$ $r_0 \leq_{\mathbb{R}} r_2$	$r_2 <_{\mathbb{R}} r_0$
$3rd$	$r_2 <_{\mathbb{R}} r_0$ $r_3 <_{\mathbb{R}} r_0$	$r_2 - r_0 \cdots r_1$ $r_2 \cdots r_{3,4,5}$	$r_2 \leq_{\mathbb{R}} r_3$	$r_2 \leq_{\mathbb{R}} r_3$	$r_3 <_{\mathbb{R}} r_2$
$4th$	$r_2 <_{\mathbb{R}} r_0$ $r_3 <_{\mathbb{R}} r_0$ $r_3 <_{\mathbb{R}} r_2$	$r_3 - r_2 - r_0 \cdots r_1$ $r_3 \cdots r_{4,5}$	$r_3 \leq_{\mathbb{R}} r_1$	$r_3 <_{\mathbb{R}} r_2$ $r_2 <_{\mathbb{R}} r_0$ $r_0 \leq_{\mathbb{R}} r_1$	$r_1 <_{\mathbb{R}} r_0$
$5th$	$r_1 <_{\mathbb{R}} r_0$ $r_2 <_{\mathbb{R}} r_0$ $r_3 <_{\mathbb{R}} r_0$ $r_4 <_{\mathbb{R}} r_2$	$r_1 - r_0$ $r_1 \cdots r_{2,3,4,5}$	$r_1 \leq_{\mathbb{R}} r_4$	$r_1 \leq_{\mathbb{R}} r_4$	$r_4 <_{\mathbb{R}} r_1$
$6th$	$r_1 <_{\mathbb{R}} r_0$ $r_2 <_{\mathbb{R}} r_0$ $r_3 <_{\mathbb{R}} r_0$ $r_4 <_{\mathbb{R}} r_2$ $r_4 <_{\mathbb{R}} r_1$	$r_4 - r_1 - r_0$ $r_1 \cdots r_{2,3}$ $r_4 \cdots r_5$	$\ldots$	$\ldots$	$\ldots$

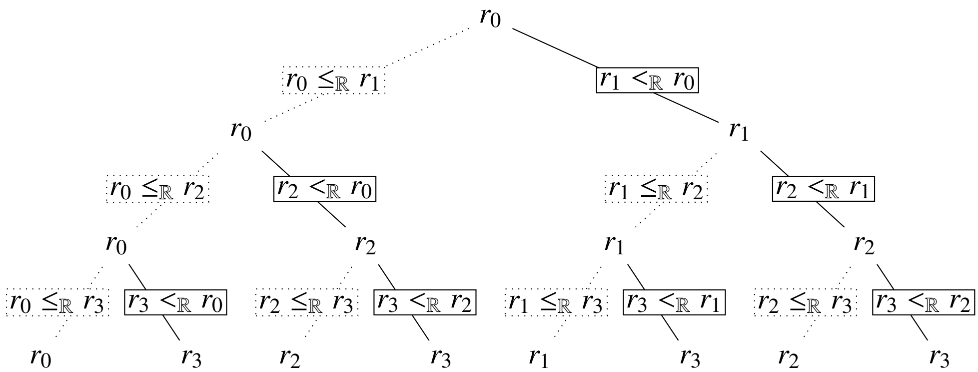
Iter: *the iteration represented by the current row;* **State:** *the existential disjuncts witnessed by the state;* **Least element:** *the least element yielded by the Least Element Lemma;* **Used:** *a falsifiable consequence of the Least Element Lemma used in the proof;* **Deduced from:** *what we deduced the falsifiable consequence from;* **Discovered:** *the existential assumption we found a witness of.*

we encountered along the path, an instance we did not have a witness for. This means that in the next computation, when we arrive at the node corresponding to that EM_1 instance, instead of taking the left path as we did previously (since the state did not have a witness for that instance), we take the right path, because this time we do have a witness (since we just extended the state with it). Therefore, each time we backtrack, the computation path ends with a leaf that is more to the right in Figure 5.4. This gives a bound to the number of backtrackings, namely $2^n - 1$.

This is very different from what one could expect by a superficial look at the proof of the Least Element Lemma. Indeed, if we ignore the undecidability of the order on the reals,

Figure 5.4: The computation tree of the least element for $n = 3$

.



Each path represents a possible computation, proceeding from root to leaf, where non-leaf nodes are the current least element candidates and the leaf is the final result. Each branching corresponds to an EM_1 *instance, where the left branch is taken when we guess that the universal disjunct holds for lack of information and the right branch is taken when the state contains the relevant witness.*

this simple and very natural proof seems to be quite efficient, since its complexity is linear in n. However, its interactive interpretation has exponential complexity. This can be seen in the computation tree too: a single computation correspond to a path and paths have length n. On the other hand, since we have backtracking, in the worst case we may have to perform every possible computation. Naturally, the real situation is different since the order on the reals *is* undecidable and thus an actual comparison is impossible.

Moreover, while in the worst case the interactive interpretation needs a time that is exponential in n, in general it is hard to estimate the amount of backtracking that will be actually performed, for two different reasons.

- The first one is that the actual order of $r_0, \ldots, r_n$ affects heavily the operation of the algorithm. Indeed, assume that r_0 is the least element: the interactive interpretation only performs n dummy comparisons and immediately returns a least element candidate that, in this case, is the actual least element, so no backtracking can ensue later.

- The second reason is that the backtracking is controlled by how the least element candidate returned by the interactive interpretation is used. It is possible for the interactive

interpretation to return a candidate that is not a least element, but such that its use in an outer proof is does not cause backtracking. In other words, we only need to compute a least element candidate that is good enough instead of the correct one and this can translate to a faster computation, again depending on the situation.

In a sense, the second reason explains also how the interactive interpretation is effective even if an certainly correct least element cannot be found effectively.

5.3.2 The Whole Proof Is Relevant

In Remark 3, we said that proofs contains both an algorithm (which may be trivial if no information is being computed) and the proof of its correctness. This is also the case when we consider the computational content of a proof in the BHK interpretation: we can separate the part that computes values and such (the informative computation) from the part that computes the evidence showing that the values are correct (the correctness computation). The correctness computation does not affect the result of the informative computation and can be safely discarded when we are only interested in algorithm extraction.

This is not the case for the computational content in the interactive interpretation. Here the correctness part of the computation affects the backtracking, which affects the state, which in turn affects the informative part of computation and thus the computed values. Therefore, in interactive realizability both parts of the proof interact to produce the final result.

We have already seen an example of this interaction. In the second iteration we chose r_3 as the least element and then we tried to instance $r_3 \leq_{\mathbb{R}} r_2$. Then we realized that $r_3 \leq_{\mathbb{R}} r_2$ is false and that we had made a wrong assumption somewhere. However $r_3 \leq_{\mathbb{R}} r_2$ is not one of the universal disjunct which we assumed by lack of information. Therefore we have to look at the proof in order to find out which universal disjuncts we needed to deduce $r_3 \leq_{\mathbb{R}} r_2$ and to compute the witness which we need to extend the state. This shows that in the interactive interpretation we cannot forget how we proved the correctness of our computations.

5.4 The Real Plane

In this section we introduce the real plane, points, lines and some relations between them. We use elementary analytic geometry: points are represented by coordinates, lines by equations and proofs are mostly computations with real numbers.

We represent a point as a pair of real numbers, its coordinates. Formally we can say that a point is just a natural number i and that there is a primitive recursive function mapping indexes into pairs of real numbers. As we did for real numbers, in order to improve readability we add some sugar to the notation and use the metavariables P, Q, R, S for arithmetic terms used as indexes of points. When we use the index of a point both as a number and as a point, we write it as i in the first case and as P_i in the second. We write the coordinates of a point P as (x_P, y_P) and of a point P_i as (x_i, y_i).

A line passing through two points PQ is written as PQ. The order of the points induces an orientation on the line.

Before proceeding we need to introduce further infrastructure for the real numbers.

5.4.1 Operations on Real Numbers

Any rational number q can be embedded in our coding of the real numbers: indeed we can represent q as a real number by taking the nested interval sequence with the lower and higher extremes constantly equal to q. In particular we assume that there is an index $0_{\mathbb{R}}$ such that f^+ and f^- are constantly zero.

We need to introduce the addition, subtraction and multiplication operations on the reals. In order to do this formally, we need to assume that for each pair of indexes i and j of real numbers, there is an index k which correspond to the nested interval sequence that is the result of their sum, difference or product. Again, instead of writing the index k, we use the usual syntax $r_i +_{\mathbb{R}} r_j$ for the sum, $r_i -_{\mathbb{R}} r_j$ for the difference and $r_i \cdot_{\mathbb{R}} r_j$ for the product.

Now we define the actual sequences that represent the result of each operation and show that they satisfy the real number condition (5.2).

We define addition on the nested interval sequences as:

$$(r_i +_{\mathbb{R}} r_j)^+(k) \equiv r_i^+(k+1) +_{\mathbb{Q}} r_j^+(k+1),$$
$$(r_i +_{\mathbb{R}} r_j)^-(k) \equiv r_i^-(k+1) +_{\mathbb{Q}} r_j^-(k+1).$$

It is immediate to check that the the sequences are a sequence of nested intervals; we only check that they converge with the required speed, which is the condition that requires the use of $k + 1$ in the previous definition:

$$\begin{aligned}
&(r_i +_{\mathbb{R}} r_j)^+(k) -_{\mathbb{Q}} (r_i +_{\mathbb{R}} r_j)^-(k) =_{\mathbb{Q}} \\
=_{\mathbb{Q}}\ &(r_i^+(k+1) +_{\mathbb{Q}} r_j^+(k+1)) -_{\mathbb{Q}} (r_i^-(k+1) +_{\mathbb{Q}} r_j^-(k+1)) =_{\mathbb{Q}} \\
=_{\mathbb{Q}}\ &(r_i^+(k+1) -_{\mathbb{Q}} r_i^-(k+1)) +_{\mathbb{Q}} (r_j^+(k+1) -_{\mathbb{Q}} r_j^-(k+1)) \leq_{\mathbb{Q}} \\
\leq_{\mathbb{Q}}\ &2^{-k+1} +_{\mathbb{Q}} 2^{-k+1} =_{\mathbb{Q}} 2^{-k}.
\end{aligned}$$

We can define the difference by combining the sum and the opposite, which is defined as:

$$\begin{aligned}
(-r)^+(k) &\equiv -_{\mathbb{Q}} r^-(k), \\
(-r)^-(k) &\equiv -_{\mathbb{Q}} r^+(k).
\end{aligned}$$

Defining the product is slightly more complicated. For simplicity we only show the case when the extreme of the intervals are always positive.

So let r_i^+, r_i^-, r_j^+ and r_j^- be sequences of positive rational numbers. We define their product as:

$$\begin{aligned}
(r_i \cdot_{\mathbb{R}} r_j)^+(k) &\equiv r_i^+(l) \cdot_{\mathbb{Q}} r_j^+(l), \\
(r_i \cdot_{\mathbb{R}} r_j)^-(k) &\equiv r_i^-(l) \cdot_{\mathbb{Q}} r_j^-(l),
\end{aligned}$$

where l depends on k. In order to determine l we consider the convergence condition and look for the smallest l that satisfies it:

$$(r_i \cdot_{\mathbb{R}} r_j)^+(k) -_{\mathbb{Q}} (r_i \cdot_{\mathbb{R}} r_j)^-(k) \leq_{\mathbb{Q}} 2^{-k}.$$

We begin by finding a simple upper bound for the left-hand side:

$$\begin{aligned}
&(r_i \cdot_{\mathbb{R}} r_j)^+(k) -_{\mathbb{Q}} (r_i \cdot_{\mathbb{R}} r_j)^-(k) =_{\mathbb{Q}} \\
=_{\mathbb{Q}}\ &r_i^+(l) \cdot_{\mathbb{Q}} r_j^+(l) -_{\mathbb{Q}} r_i^-(l) \cdot_{\mathbb{Q}} r_j^-(l) =_{\mathbb{Q}} \\
=_{\mathbb{Q}}\ &r_i^+(l) \cdot_{\mathbb{Q}} r_j^+(l) -_{\mathbb{Q}} r_i^-(l) \cdot_{\mathbb{Q}} r_j^-(l) -_{\mathbb{Q}} r_i^+(l) \cdot_{\mathbb{Q}} r_j^-(l) +_{\mathbb{Q}} r_i^+(l) \cdot_{\mathbb{Q}} r_j^-(l) =_{\mathbb{Q}} \\
=_{\mathbb{Q}}\ &r_i^+(l) \cdot_{\mathbb{Q}} (r_j^+(l) -_{\mathbb{Q}} r_j^-(l)) +_{\mathbb{Q}} r_j^-(l) \cdot_{\mathbb{Q}} (r_j^+(l) -_{\mathbb{Q}} r_j^-(l)) \leq_{\mathbb{Q}} \\
\leq_{\mathbb{Q}}\ &r_i^+(l) 2^{-l} +_{\mathbb{Q}} r_j^-(l) 2^{-l} =_{\mathbb{Q}} (r_i^+(l) +_{\mathbb{Q}} r_j^-(l)) 2^{-l}.
\end{aligned}$$

Thus the convergence condition is satisfied when:

$$(r_i^+(l) +_{\mathbb{Q}} r_j^-(l))2^{-l} \leq_{\mathbb{Q}} 2^{-k}.$$

We define k as the smallest natural number satisfying the previous inequality.

5.4.2 The Left and Right Predicates

In order to write the formal statement of the Convex Angle Theorem, we need a way to determine the position of a point with respect to a line.

First of all consider two points P and Q. We can write the equation that a point R has to satisfy to be on the line going through them:

$$(x_Q - x_P)(y_R - y_P) - (x_R - x_P)(y_Q - y_P) =_{\mathbb{R}} 0_{\mathbb{R}}. \tag{5.7}$$

If the left-hand side is zero then R is on the same line with P and Q. When left-hand side is not zero, we can use its sign to distinguish which side of PQ R is on. We call these sides left and right. We write $\mathsf{left}(P, Q, R)$ (resp. $\mathsf{right}(P, Q, R)$) and we say that R is to the *left* (resp. *right*) of the line passing through the points P and Q when

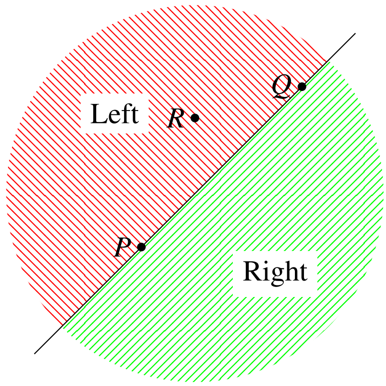
$$\begin{aligned}\mathsf{left}(P, Q, R) &\equiv (x_Q - x_P)(y_R - y_P) - (x_R - x_P)(y_Q - y_P) >_{\mathbb{R}} 0_{\mathbb{R}},\\ \mathsf{right}(P, Q, R) &\equiv (x_Q - x_P)(y_R - y_P) - (x_R - x_P)(y_Q - y_P) <_{\mathbb{R}} 0_{\mathbb{R}},\end{aligned}$$

as seen in Figure 5.5. A few remarks on this definition:

- left and right are positively decidable, since they are defined by means of $<_{\mathbb{R}}$;
- since the definitions of left and right are almost the same and only the direction of the inequality changes, R is to the left of PQ if and only if Q is to the right of PQ;
- the left side of PQ corresponds to the right side of QP and the other way around, so the order of the points is significant;
- the left-hand side of (5.7) can also be thought as the scalar product of $(-(y_Q - y_P), x_Q - x_P)$, the orthogonal of the vector from P to Q, and $(x_R - x_P, y_R - y_P)$, the vector from P to R.

We say that P is *above* Q if $y_P \geq_{\mathbb{R}} y_Q$ and that R is *below* Q when $y_R \leq_{\mathbb{R}} y_Q$.

Figure 5.5: R is to the left of PQ.



5.5 The Geometric Part of the Proof

Now we are ready to present the rest of the proof of the main statement. We divide the proof in two parts, the first given as a lemma. Since these proofs are more complex, for reason of readability and space we will not be as formal as we have we have been until now.

From this point onward we assume that no three points are on the same line, formally:

$$\forall P, Q, R.\ \mathsf{left}(P, Q, R) \vee \mathsf{right}(P, Q, R). \tag{5.8}$$

This a strong assumption, even more so because we require this to hold constructively: since left and right are Σ^0_1 formulas defined with $\leq_{\mathbb{R}}$, we assume that we have an effective map that given three points yields the precision we need to reach in order to check that R is not on the line PQ. In other words, we are assuming that we have a procedure that effectively decides instances of the left and right predicates. The effective computation we extract uses this procedure as a parameter.

A further consequence is that all points must be distinct: when $x_P =_{\mathbb{R}} x_Q$ and $y_P =_{\mathbb{R}} y_Q$, the left-hand side in (5.7) is always zero for any R.

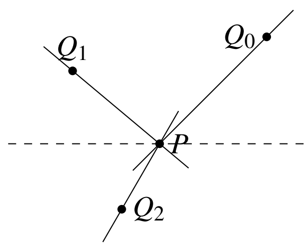
In the next lemma the points Q_0, Q_1, Q_2 are three generic points, that is, Q_i is not necessarily the point indexed by the natural number i. Moreover we assume that the index i in Q_i is interpreted up to congruence modulo 3 and thus always falls in $\{0, 1, 2\}$. For instance,

when we write Q_4, we actually mean Q_1. We write the coordinates of Q_i as (x_i, y_i), with the same conventions for the index. We prove that when three points are one to the left of the other with respect to a central one, one of them is necessarily lower than the central point, as shown in Figure 5.6.

Lemma 8 (Three points)**.** *Assume* (5.8) *and let P, Q_0, Q_1 and Q_2 be four points in the real plane such that Q_{i+1} is to the left (resp. right) of PQ_i for any $i < 3$. Then at least one of Q_0, Q_1, Q_2 is strictly below P. Formally:*

$$\forall P, Q_0, Q_1, Q_2. (\forall i < 3.\ \mathsf{left}(P, Q_i, Q_{i+1})) \to \exists i < 3.\ y_i <_{\mathbb{R}} y_P.$$

Figure 5.6: The three points lemma when Q_2 is the point below P.



Classical proof. Without loss of generality we can assume that the coordinates of P are $(0_{\mathbb{R}}, 0_{\mathbb{R}})$. Then, unfolding the definition of left, the hypothesis on the points can be written as:

$$\forall i < 3.\ x_i y_{i+1} - x_{i+1} y_i >_{\mathbb{R}} 0_{\mathbb{R}},$$

The first step is showing that there at least two points whose vertical coordinate is not zero. This follows from the fact that if, for some $i < 3$, $y_i = 0_{\mathbb{R}}$ then $y_{i+1} \neq 0_{\mathbb{R}}$ and $y_{i-1} \neq 0_{\mathbb{R}}$. This is the case since if $y_i = 0_{\mathbb{R}}$ then

$$x_{i-1} y_i - x_i y_{i-1} = -x_i y_{i-1} >_{\mathbb{R}} 0_{\mathbb{R}},$$
$$x_i y_{i+1} - x_{i+1} y_i = x_i y_{i+1} >_{\mathbb{R}} 0_{\mathbb{R}},$$

and then $y_{i+1} \neq 0_{\mathbb{R}}$ and $y_{i-1} \neq 0_{\mathbb{R}}$. Then we can assume that y_{i+1} and y_{i-1} are not zero. If either of them is negative we can conclude.

Otherwise they are both positive. We show that, in this case, y_i is negative. By hypothesis we know that:

$$x_{i-1}y_i - x_iy_{i-1} >_{\mathbb{R}} 0_{\mathbb{R}},$$
$$x_iy_{i+1} - x_{i+1}y_i >_{\mathbb{R}} 0_{\mathbb{R}}.$$

Since y_{i+1} and y_{i-1} are positive we can multiply the previous inequalities:

$$y_{i+1}(x_{i-1}y_i - x_iy_{i-1}) >_{\mathbb{R}} 0_{\mathbb{R}},$$
$$y_{i-1}(x_iy_{i+1} - x_{i+1}y_i) >_{\mathbb{R}} 0_{\mathbb{R}}.$$

By adding them together we get:

$$y_i(x_{i-1}y_{i+1} - x_{i+1}y_{i-1}) > 0.$$

Since the term in parenthesis is negative by hypothesis, y_i must be too. Since Q, R and S are to the right of each other with respect to P if and only if S, R and Q are to the left of each other with respect to P, the proof for right is basically the same. □

The previous proof can be made constructive. Since the proofs are very similar, we give the intuitionistic proof without explaining the how we obtained it from the classical one. The main difference is that, in the intuitionistic proof, we work directly on the rational intervals approximating the coordinates of the points and thus we use rational arithmetic which is decidable. For any precision$k$, let $X_i(k)$ and $Y_i(k)$ be the closed rational intervals $[x_i^-(k), x_i^+]$ and $[y_i^-(k), y_i^+]$ respectively. We write $q_i \in X_i(k)$ as a compact notation for $x_i^- \leq_{\mathbb{Q}} x_i \wedge x_i \leq_{\mathbb{Q}} x_i^+$.

Intuitionistic proof. Without loss of generality we can assume that the coordinates of P are $(0_{\mathbb{R}}, 0_{\mathbb{R}})$. Then, unfolding the definition of left, the hypothesis on the points can be written as:

$$\forall i < 3.\ x_iy_{i+1} - x_{i+1}y_i >_{\mathbb{R}} 0_{\mathbb{R}}.$$

By unfolding the definition of real numbers as sequences of nested intervals and these of the operations on real numbers, we can compute some precision k such that:

$$\forall i < 3.\ \forall q_i \in X_i(k), y_i \in Y_i(k), q_{i+1} \in X_{i+1}(k), y_{i+1} \in Y_{i+1}(k).\ q_iy_{i+1} - q_{i+1}y_i >_{\mathbb{Q}} 0_{\mathbb{Q}}. \quad (5.9)$$

The first step is showing that there at least two points whose vertical coordinate is not zero. In order to show this, assume that for some $i < 3$ we have $0_{\mathbb{Q}} \in Y_i(k)$. Then we can take $p_i = 0_{\mathbb{Q}}$ in (5.9), for $i-1$ and i:

$$\forall q_{i-1} \in X_{i-1}(k), q_i \in X_i(k), p_{i-1} \in Y_{i-1}(k).\ q_{i-1}p_i - q_i p_{i-1} = -q_i p_{i-1} >_{\mathbb{Q}} 0_{\mathbb{Q}},$$
$$\forall q_i \in X_i(k), q_{i+1} \in X_{i+1}(k), p_{i+1} \in Y_{i+1}(k).\ q_{i-1}p_i - q_i p_{i-1} q_i p_{i+1} - q_{i+1} p_i = q_i p_{i+1} >_{\mathbb{Q}} 0_{\mathbb{Q}}.$$

Therefore, for $j \in \{i-1, i+1\}$, $0_{\mathbb{Q}} \notin Y_j(k)$ and, since $Y_j(k)$ is an interval, it must be either completely positive or completely negative, namely, either $x_j^- >_{\mathbb{Q}} 0_{\mathbb{Q}}$ or $x_j^+ <_{\mathbb{Q}} 0_{\mathbb{Q}}$. If either one is completely negative then we have the conclusion.

Otherwise they are both completely positive and we show that, in this case, $Y_i(k)$ is completely negative. For all $q_j \in X_j(k)$ and all $p_j \in Y_j(k)$ with $j \in \{0, 1, 2\}$; we know by hypothesis that:

$$q_{i-1}p_i - q_i p_{i-1} >_{\mathbb{Q}} 0_{\mathbb{Q}},$$
$$q_i p_{i+1} - q_{i+1} p_i >_{\mathbb{Q}} 0_{\mathbb{Q}}.$$

Since p_{i+1} and p_{i-1} are positive we can multiply the previous inequalities:

$$p_{i+1}(q_{i-1}p_i - q_i p_{i-1}) >_{\mathbb{Q}} 0_{\mathbb{Q}},$$
$$p_{i-1}(q_i p_{i+1} - q_{i+1} p_i) >_{\mathbb{Q}} 0_{\mathbb{Q}}.$$

By adding them together we get:

$$p_i(q_{i-1}p_{i+1} - q_{i+1}p_{i-1}) >_{\mathbb{Q}} 0_{\mathbb{Q}}.$$

Since the term in parenthesis is negative by hypothesis, p_i must be too, for all $p_i \in Y_i$.

Since Q, R and S are to the right of each other with respect to P if and only if S, R and Q are to the left of each other with respect to P, the proof for right is basically the same. □

We can now prove the main statement.

Theorem 6 (Convex Angle)**.** *Assume* (5.8). *For any $n \geq 2$, we can select three points P, Q and R from $\{P_0, \ldots, P_n\}$ such that all the remaining points fall in the angle $\widehat{QPR}$, that is, all points are to the left of PQ and to the right of PR.*

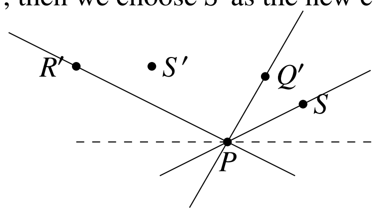
$$\forall n \geq 2.\ \exists i, j, k \leq n.\ \forall l \leq n.\ l \neq i \wedge (l \neq j \rightarrow \mathsf{left}(P_i, P_j, P_l)) \wedge (l \neq k \rightarrow \mathsf{right}(P_i, P_k, P_l)).$$

Classical proof. Let P be the point with the least vertical coordinate and choose other two points Q' and R', which are our candidates for Q and R respectively. We want all points except P to be to the left of PQ and to the right of PR. If Q' is to the left of PR', we swap Q' and R'. Thus we know that Q' is to the right of PR' and R' is to the left of PQ'.

Now consider any point S except P, Q' and R'. We have four cases:

- If S is to the left of PQ' and if it is to the right of PR', then we keep Q' and R' as candidates for Q and R.

- If S is to the right of PQ', then we choose S as the new candidate for Q.

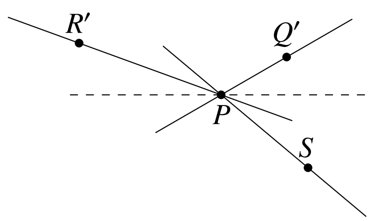


  Clearly Q' is to the left of PS. Moreover, any other point S', which we already checked to be to the left of PQ', is to the left of PS too. This is a consequence of (5.8) and Lemma 8.

  Indeed, from (5.8), we know that S' is either to the left or to the right of PS. We already know that S is to the right of PQ' and Q' is to the right of PS'. If S' were to the right of PS, then by Lemma 8, one of Q', S or S' would have be strictly lower than P which would be a contradiction, since P is the lowest point. Thus S' is to the left of PS.

- Symmetrically, if S is to the left of PR', then we choose S as the new candidate for R.

- We shot that S cannot be to the right of PQ' and to the left of PR':



  If this were the case, Q' would be to the left of PS and S would be to the left of PR'. Since we know that R' is to the left of PQ', by Lemma 8, one of S, Q' or R' would be

strictly lower than P. This is a contradiction, since P is the lowest point by the Least Element Lemma.

We repeat this procedure for all the points except P, Q' and R' and we find the required points Q and R. □

For convenience we have written the proof as an iterative algorithm. The proof is actually by induction on a slightly stronger version of the final statement, that adds the requirement for P to be lower than all the other points.

5.6 The Interactive Interpretation

Before studying the interactive interpretation of the whole proof of the Convex Angle Theorem along with its lemmas, we need understand their computational significance. Thus we stop for a moment and recall some general considerations on the computational meaning of formulas in the BHK interpretation and, more specifically, in the Curry-Howard correspondence.

As a consequence of the proof-as-programs and formulas-as-types interpretation, the conclusion of a proof (that is, the statement it proves) can be thought of as the specification of the program representing the proof.

5.6.1 Subroutines, arguments and effective computations

In order to understand how the interactive interpretation works, it is important to distinguish computations that can be carried out effectively from those that cannot. Consider a proof of a statement of the form:

$$\forall x.\ \exists y.\ A. \tag{5.10}$$

If we read the previous formula as a specification, it calls for a program that describes a function, a subroutine. It takes a natural number as an argument named x and returns a pair containing a natural number y and a program/proof that y satisfies A. More generally, statements in mathematics have the following form:

$$\forall x_1, \ldots, x_n.\ A_1 \wedge \cdots \wedge A_m \to A.$$

This can be again seen as the specification of a subroutine, taking n natural numbers and m programs as arguments. From this point of view, it becomes clear that such a program is not computing anything, at least in itself. An effective computation can only start once the subroutine is applied to an argument.

All of our theorems begin with universal quantifications and implications, that is, they are specification for programs that code functions with arguments. Thus, in order to have an actual computation we have to provide the program with the required arguments.

5.6.2 The Interactive Interpretation of the Whole Proof

We can now explain the interactive interpretation of the whole proof, composed of the two lemmas and the final algorithmic proof. We focus on the interaction between these parts without analyzing each part in detail as we have done for the Least Element Lemma.

We start by considering the statement of the Convex Angle Theorem. Assume that we are given a natural number n. In the proof we work with the first $n+1$ points of the enumeration.

The proof is an iterative procedure to select P, Q and R satisfying the following *bounding condition*:

$$\forall l \leq n.\, l \neq i \wedge (l \neq j \rightarrow \mathsf{left}(P_i, P_j, P_l)) \wedge (l \neq k \rightarrow \mathsf{right}(P_i, P_k, P_l)). \tag{5.11}$$

The bounding condition specifies an informative computation, since left and right are defined by means of $<_{\mathbb{R}}$, which is an existential quantification. Thus its proofs computes some witnesses, namely the precision of the comparisons we need to check that the bounding condition holds. While are mainly interested in the choice of the points P, Q and R and not in the information needed to prove the bounding condition itself, the precision of the computation provided by (5.11) is actually used in interactive interpretation since it can cause backtracking.

We claim that this bounding condition specifies an effective computation. First of all, the outer universal quantification is bounded, thus, in order to compute the condition, we have to compute the body of the quantification $n+1$ times. The same holds for the conjunctions. Thus the effectiveness of the whole condition follows from the effectiveness of the conjuncts. The implications are effective: their only argument, the proof of the antecedent, is arithmetical atomic, hence irrelevant, thus the computations they specify must be constant functions. Therefore, we can effectively compute them by applying them to any single

argument. Finally their consequents specify effective computations, thanks to (5.8), the assumption that no three points are on the same line. Thus, proofs of the bounding condition describe effective computations.

Now we can start following the proof. In the beginning, the lowest point P is selected using the Least Element Lemma on the vertical coordinate. Consider the statement of the Least Element Lemma:

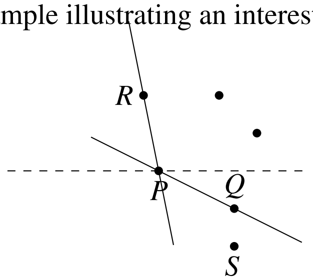
$$\forall n.\ \exists i \leq n.\ \forall j \leq n.\ r_i \leq_{\mathbb{R}} r_j.$$

As a specification, it calls for a program that, given n, yields the value i and the correctness computation that checks that i is the least element. Since the correctness computation cannot be carried out effectively (it is negatively decidable), the interactive interpretation computes a trivial least element the first time. If later in the proof we happen to partially compute the correctness computation, then we may discover new information and backtrack again to the least element computation. Since the Least Element Lemma does not necessarily return a least element, but only a least element candidate, P is not the lowest point either, but just a lowest point candidate.

The function of Lemma 8 is to prove that some point is strictly lower than P, thus producing a contradiction. In the classical proof this ensures that undesirable situations never happen. In the interactive interpretation however, since P is not necessarily the lowest point, no contradiction occurs. Instead, what happens is that we actually are in one of the cases we had excluded in the classical proof. At this point, in order to deduce the contradictory statement, we have partially computed the correctness computation returned by the Least Element Lemma and thus discovered which assumption was incorrectly guessed. We compute the relevant witness and extend the state accordingly. Then we compute a new lowest point candidate and continue again following the proof of the Convex Angle Theorem until either we can satisfy its conclusion or we backtrack again.

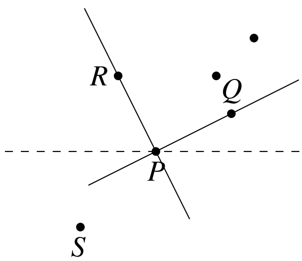
We use Lemma 8 in two places in the proof of the Convex Angle Theorem. The first use takes place when, while iterating on the points, we discover that the bounding condition fails for some S and we choose S as the new candidate for Q or R. We use Lemma 8 to show that this choice satisfies the bounding condition for all the previous points we iterated over until now. More precisely we use Lemma 8 to prove that, if the bounding conditions fails for S, then one of Q, R or S is strictly lower than P. As we described previously, this in turn starts

the backtracking. Here is an example illustrating an interesting situation:



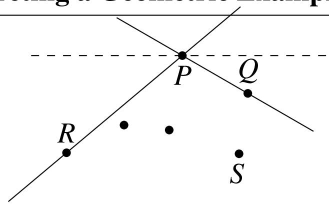
Here S is to the right of PQ, so we replace Q with S. From the picture we see that S is actually strictly lower than P. On the other hand the bounding condition is satisfied by taking S as P. In this situation, do we backtrack or not? We know that we can find a better candidate for the lowest point, but P seems to be good enough already, so there is no real need for a better candidate. Both options are sound and the choice depends on the exact formalization of the proof and the exact sequences of rationals representing the vertical coordinate of the points in question.

We also use Lemma 8 to claim that the bounding condition cannot fail because S is both to the right of PQ and to the left of PR:



This case was excluded completely in the classical proof, since it always leads to contradiction. When it occurs in the interactive interpretation, we backtrack for sure since the bounding condition cannot be satisfied. More precisely, in this case Lemma 8 proves that one of Q, R or S is strictly lower than P. Therefore, in order to get the contradiction, we instance the assumptions $y_P \leq_{\mathbb{R}} y_Q$, $y_P \leq_{\mathbb{R}} y_R$ and $y_P \leq_{\mathbb{R}} y_S$ with enough precision to falsify at least one of them.

As a last example, consider a situation where the state is empty and thus P is simply the first point in the enumeration. Assume that the points are arranged as shown:



Since the bounding condition is satisfied immediately, we never need to use Lemma 8. Thus backtracking never ensues. This mean that P, while certainly not the lowest point, is a good enough candidate and we do not need another one. This is one of the cases we mentioned where the interactive interpretation produces a fast computation, since the lowest point is only computed once and the proof ends with no backtracking. This shows how the behavior of interactive interpretation of the Least Element Lemma depends heavily on the final statement of the proof.

Appendix A

Appendix

A.1 Additional Reductions

In this section we list the standard reductions (given in [20]) that we did not need in order to prove Lemma 4. The main reason is that in the proof we are only concerned with principal branches and some reductions only affect non-principal branches. We list them for completeness.

A.1.1 Permutative Reductions

We saw how the proper reductions are performed when the conclusion of an introduction rule instance is the major premiss of a elimination rule instance. The situation becomes less straightforward when the conclusion of an introduction rule instance α is a minor premiss of an $\vee$E or $\exists$E rule instance γ whose conclusion is in turn the major premiss of an elimination rule instance β. Also in this case the formula introduced by α is eliminated by β, but we cannot apply a proper reduction since γ is in the way. What we can do is to rearrange the derivation by moving β above (or "inside") γ, so that β is immediately below α and we can apply the suitable proper reduction. Therefore we have two *permutative reductions*, depending on whether γ is an instance of the $\vee$E or the $\exists$E rule. Note that repeated application of the permutative reductions allows us to apply a proper reduction even when there is more than one instance of the $\vee$E or the $\exists$E between α and β. Thus they can be thought of as auxiliary reductions that can eventually enable a suitable proper reduction. They are listed

in Figure A.1.

Figure A.1: The permutative reductions.

$\vee$–perm

$$
{*}\mathrm{E}\;\dfrac{\vee\mathrm{E}\;\dfrac{\dfrac{\Pi_1}{A_1 \vee A_2} \qquad \dfrac{\begin{matrix}[A_1]^{\gamma}\\ \Pi_2\end{matrix}}{B} \qquad \dfrac{\begin{matrix}[A_2]^{\gamma}\\ \Pi_3\end{matrix}}{B}}{B}\,\gamma \qquad \Pi_4}{C}\,\beta
$$

$$\downarrow$$

$$
\vee\mathrm{E}\;\dfrac{\dfrac{\Pi_1}{A_1 \vee A_2} \qquad {*}\mathrm{E}\;\dfrac{\dfrac{\begin{matrix}[A_1]^{\gamma}\\ \Pi_2\end{matrix}}{B} \qquad \Pi_4}{C}\,\beta \qquad {*}\mathrm{E}\;\dfrac{\dfrac{\begin{matrix}[A_2]^{\gamma}\\ \Pi_3\end{matrix}}{B} \qquad \Pi_4}{C}\,\beta}{C}\,\gamma
$$

$\exists$–perm

$$
{*}\mathrm{E}\;\dfrac{\exists\mathrm{E}\;\dfrac{\dfrac{\Pi_1}{\exists x.\, A} \qquad \dfrac{\begin{matrix}[A[x := y]]^{\gamma}\\ \Pi_2\end{matrix}}{B}}{B}\,\gamma \qquad \Pi_4}{C}\,\beta
$$

$$\downarrow$$

$$
\exists\mathrm{E}\;\dfrac{\dfrac{\Pi_1}{\exists x.\, A} \qquad {*}\mathrm{E}\;\dfrac{\dfrac{\begin{matrix}[A[x := y]]^{\gamma}\\ \Pi_2\end{matrix}}{B} \qquad \Pi_4}{C}\,\beta}{C}\,\gamma
$$

$*$ is any logical connective: $\wedge, \vee, \rightarrow, \forall, \exists$;

Π_4 stands for the subderivations of the minor premisses (if any) of β.

A.1.2 Immediate Simplifications

Consider a different kind of avoidable complexity in derivation: an instance α of the $\vee$E or the $\exists$E rule such that the one of its minor premisses $\mathfrak{a}$ is derived without using the assumption discharged by α. More precisely this means that no occurrence of the assumption discharged by α appears in the subderivation of $\mathfrak{a}$. Whenever this is the case we say that α is *redundant* since we do not need the assumptions it provides in order to prove its conclusion. There are two reductions, called *immediate simplifications*, depending on whether α is an instance of the $\vee$E or the $\exists$E rule. They are listed in Figure A.2.

Figure A.2: The immediate simplifications.

$\vee-$simpl

$$\vee\mathrm{E}\ \dfrac{A_1 \vee A_2 \qquad \dfrac{\begin{matrix}[A_1]^{\alpha}\\ \Pi_1\end{matrix}}{C} \qquad \dfrac{\begin{matrix}[A_2]^{\alpha}\\ \Pi_2\end{matrix}}{C}}{C}\ \alpha \qquad \leadsto \qquad \dfrac{\Pi_i}{C}$$

if Π_i does not depend on A_i

$\exists-$simpl

$$\exists\mathrm{E}\ \dfrac{\exists x.\, A \qquad \dfrac{\begin{matrix}[A[x := y]]^{\alpha}\\ \Pi\end{matrix}}{B}}{B}\ \alpha \qquad \leadsto \qquad \dfrac{\Pi}{C}$$

A.2 Witness Reduction in Two Steps

Instead of giving the single reduction Wit$-$red, we can split it in two distinct reductions, one that looks for counterexamples and eliminates occurrences of the open assumptions of the EM_1 rule and one that eliminates instances of the EM_1 rule when their conclusion can be derived without the universal or existential assumption.

A.2.1 A Lighter Witness Extracting Reduction

More precisely consider an instance α of the EM_1 rule for the quantifier-free formula A:

$$\mathsf{EM}_1\ \dfrac{\dfrac{\begin{array}{c}[\forall x.\,A]^\alpha\\ \Pi_1\end{array}}{C} \qquad \dfrac{\begin{array}{c}[\neg A[x:=y]]^\alpha\\ \Pi_2\end{array}}{C}}{C}\ \alpha$$

Let $\mathfrak{a}$ be an occurrence of the assumption $\forall x.\, A$ discharged by α in Π_1 In order to be able to perform the reduction we assume the following:

- $\mathfrak{a}$ is the premiss of a $\forall$E instance β,
- the conclusion of β is the occurrence of a closed formula.

Let $\mathfrak{b}$ be the conclusion of β. $\mathfrak{b}$ is an occurrence of the closed quantifier-free formula $A[x := t]$ for some term t. Since closed quantifier-free formula are decidable, in the reduction we can distinguish two cases, depending on whether $A[x := t]$ is true or false.

- $A[x := t]$ is true.

  Let Π_1 be a derivation of $A[x := t]$ and replace β with Π_1:

  $$\forall\mathrm{E}\ \dfrac{[\forall x.\,A]^\alpha}{\begin{array}{c}A[x:=t]\\ \vdots\end{array}} \quad\rightsquigarrow\quad \dfrac{\Pi_1}{\begin{array}{c}A[x:=t]\\ \vdots\end{array}}$$

- otherwise $\neg A[x := t]$ is true.

  Let Π_2 be a derivation of $\neg A[x := t]$ and replace all the occurrences of the assumption $\neg A[x := t]$ discharged by α in the derivation of its rightmost premiss with Π_2:

  $$\begin{array}{c}[\neg A[x:=y]]^\alpha\\ \vdots\end{array} \quad\rightsquigarrow\quad \dfrac{\Pi_2}{\begin{array}{c}\neg A[x:=t]\\ \vdots\end{array}}$$

We denote this reduction as Wit−red.

Whenever this reduction can be applied it removes one or more occurrences of one of the assumptions discharged by α. If there are no more occurrences of such assumptions in either Π_1 or Π_2 then α is redundant and can be deleted by EM_1−simpl.

A.2.2 Immediate Simplification

Redundant EM_1 rule instances can be defined and reduced in the same way as redundant $\vee$E instances. Consider an EM_1 rule instance α such that one of its premisses is derived without using the assumption discharged by α. Then we can reduce as follows:

$$\dfrac{\dfrac{\begin{matrix}[\forall x.\, A]^{\alpha} \\ \Pi_1\end{matrix}}{C} \qquad \dfrac{\begin{matrix}[\neg A[x := y]]^{\alpha} \\ \Pi_2\end{matrix}}{C}}{C}\,\alpha \quad \rightsquigarrow \quad \dfrac{\Sigma_i}{C}$$

depending on whether it is Π_1 or Π_2 that contains no occurrence of the assumption discharged by α. We denote this reduction as EM_1−simpl.

Bibliography

[1] Yohji Akama, Stefano Berardi, Susumu Hayashi, and Ulrich Kohlenbach. An arithmetical hierarchy of the law of excluded middle and related principles. In *LICS*, pages 192–201, 2004.

[2] Federico Aschieri and Stefano Berardi. Interactive learning-based realizability for Heyting arithmetic with EM_1. *Logical Methods in Computer Science*, 6(3), 2010.

[3] Federico Aschieri, Stefano Berardi, and Giovanni Birolo. Realizability and strong normalization for a Curry-Howard interpretation of $\mathsf{HA} + \mathsf{EM}_1$, 2013.

[4] Stefano Berardi and Ugo de' Liguoro. Interactive realizers. A new approach to program extraction from non constructive proofs. *ACM Transactions on Computational Logic*, 13(2):11, 2012.

[5] Stefano Berardi and Ugo de'Liguoro. A calculus of realizers for EM_1 arithmetic (extended abstract). In *CSL*, pages 215–229, 2008.

[6] Stefano Berardi and Ugo de'Liguoro. Toward the interpretation of non-constructive reasoning as non-monotonic learning. *Information and Computation*, 207(1):63–81, 2009.

[7] Stefano Berardi and Ugo de'Liguoro. Interactive Realizers and Monads. *ArXiv e-prints*, May 2010.

[8] U. Berger, H. Schwichtenberg, and W. Buchholz. Refined program extraction from classical proofs. 114:3–25, 2002.

[9] Thierry Coquand. A semantics of evidence for classical arithmetic. *The Journal of Symbolic Logic*, 60(1):325–337, 1995.

[10] Jean-Yves Girard, Yves Lafont, and Paul Taylor. *Proof and Types*. Cambridge University Press, 1988.

[11] E. Mark Gold. Limiting recursion. *The Journal of Symbolic Logic*, 30(1):28–48, 1965.

[12] Timothy G. Griffin. A formulae-as-types notion of control. In *In Conference Record of the Seventeenth Annual ACM Symposium on Principles of Programming Languages*, pages 47–58. ACM Press, 1990.

[13] W. A. Howard. *The formulae-as-types notion of construction*, pages 480–490. Academic Press, London-New York, 1980.

[14] Stephen C. Kleene. On the interpretation of intuitionistic number theory. *The Journal of Symbolic Logic*, 10(4):pp. 109–124, 1945.

[15] Georg Kreisel. *Interpretation of analysis by means of constructive functionals of finite types*. North-Holland, Amsterdam, 1959.

[16] Jean-Louis Krivine. A general storage theorem for integers in call-by-name λ-calculus. *Theoret. Comput. Sci.*, 129(1):79–94, 1994.

[17] E. Mendelson. *Introduction to Mathematical Logic: Fourth Edition*. Wadsworth & Brooks/Cole mathematics series. Chapman and Hall, 1997.

[18] Eugenio Moggi. Notions of computation and monads. *Information and Computation*, 93:55–92, 1991.

[19] Michel Parigot. $\lambda\mu$-calculus: An algorithmic interpretation of classical natural deduction. In Andrei Voronkov, editor, *Logic Programming and Automated Reasoning*, volume 624 of *Lecture Notes in Computer Science*, pages 190–201. Springer Berlin Heidelberg, 1992.

[20] Dag Prawitz. Ideas and results in proof theory. In Jens Erik Fenstad, editor, *Proceedings of the second Scandinavian Logic Symposium*, pages 235–307. North-Holland, 1971.

[21] Joseph R. Shoenfield. *Mathematical Logic*. Addison-Wesley Publishing Company, 1967.

[22] Thoralf Skolem. The foundations of elementary arithmetic. In translator Jean van Heijenoort and ed., editors, *From Frege to Gdel: A Source Book in Mathematical Logic*. Harvard Univ. Press, 1923.

[23] Morten H. Sørensen and Pawel Urzyczyn. *Lectures on the Curry-Howard Isomorphism*, volume 149 of *Studies in logic and the foundations of mathematics*. Elsevier, Amsterdam, 1 edition, September 2006.

[24] A. S. Troelstra and H. Schwichtenberg. *Basic proof theory (2nd ed.)*. Cambridge University Press, New York, NY, USA, 2000.

[25] Philip Wadler. Comprehending monads. In *Proceedings of the 1990 ACM conference on LISP and functional programming*, LFP '90, pages 61–71, 1990.

[26] Philip Wadler. The essence of functional programming. In *Proceedings of the 19th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, POPL '92, pages 1–14, New York, NY, USA, 1992. ACM.

[27] Philip Wadler. Monads and composable continuations. *LISP and Symbolic Computation*, 7:39–55, 1994. 10.1007/BF01019944.